Kasper Kiiskinen

Hallintaportaali sosiaalisen mobiilipelin ylläpitoon

Metropolia Ammattikorkeakoulu

Insinööri (AMK)

Tietotekniikan koulutusohjelma

Insinöörityö

23.4.2016

Metropolia

Tekijä Otsikko Sivumäärä Aika	Kasper Kiiskinen Hallintaportaali sosiaalisen mobiilipelin ylläpitoon 38 sivua 23.4.2016
Tutkinto	Insinööri (AMK)
Koulutusohjelma	Tietotekniikan koulutusohjelma
Suuntautumisvaihtoehto	Ohjelmistotekniikka
Ohjaajat	Lehtori Antti Laiho CTO Ville Sillanpää

Insinöörityön tarkoituksena oli kehittää Vulpine Gamesille web-sovellus, jolla voidaan seurata ja hallinnoida Last Planets -pelin pelimaailmaa. Last Planets on ilmaispelattava strateginen massiivimoninpeli IOS:lle, jossa kaikki pelaajat ovat samassa universumissa ja kehittävät omaa planeettaansa.

Hallintaportaali kehitettiin Unity3D:llä käyttäen C#-ohjelmointikieltä. Unity3D:llä voidaan kehittää sovelluksia monille eri alustoille, esimerkiksi kehittää WebGL-sovelluksia. WebGL-sovellusta voidaan käyttää selaimella, joten näin käyttäjät voivat päästä siihen käsiksi mistä vain. Autentikoituminen hallintaportaaliin toteutettiin käyttäen OAuth 2.0 -protokollaa. Portaali hakee käynnistyessään access token -avaimen Azuren Active Directoryltä käyttäen portaalin tunnuksia. Access tokenilla portaali pääsee käsiksi taustajärjestelmän APIin. Taustajärjestelmän APIn avulla portaali pystyy kommunikoimaan pelimaailman ja pelintaustajärjestelmän kanssa. Portaali ohjelmoitiin hakemaan APIlta pelimaailman uusin tilanne ja muita hyödyllisiä tietoja ja esittämään ne tämän jälkeen käyttäjälle. Näiden tietojen esittämiseen kehitettiin erilaisia toimintoja. Lämpökarttojen generointi luotiin maailman arvojen analysoimiseen ja ympyrädiagrammit luotiin hahmottamaan arvojen välisiä osuuksia.

Lopputuloksena portaalilla pystyy seuraamaan pelimaailman kehitystä, pelaajien ja liittoutumien tietoja sekä ongelmatikettejä. Portaalista pystyy myös kontrolloimaan pelimaailmaa alkeellisesti. Pelaajille pystyy esimerkiksi antamaan ja poistamaan porttikieltoja, sekä luomaan uusia tapahtumia pelimaailmaan. Hallintaportaali antaa pelinkehittäjille paljon lisäarvoa pelinkehitykseen. Lisäarvoa portaalista saadaan sillä, että siinä voidaan helposti visualisoida paljon erilaista dataa maailmasta ja pelaajista. Portaalissa pystyy tekemään lämpökarttoja, jotka auttavat pelimaailmaan seuraamisessa. Näillä työkaluilla portaalissa voidaan analysoida, miten peliä pitäisi parantaa, jotta siitä saadaan vieläkin miellyttävämpi kokemus pelaajille.

Hallintaportaaliin saatiin toteutettua ajallaan kaikki määrittelydokumentin ominaisuudet. Tämän lisäksi portaaliin saatiin myös toteutettua joitain lisäominaisuuksia, kuten ympyrädiagrammit. Portaalin jatkokehitys ja käyttöönotto alkaa lähitulevaisuudessa. Portaaliin on mahdollista lisätä uusia ominaisuuksia sitä mukaa, kun esijulkaisussa tulee esille, minkälaisille uusille työkaluille olisi käyttöä.

Avainsanat	Hallintaportaali, Unity3D, WebGL, C#, OAuth, Datan visualisointi

Author Title Number of Pages Date	Kasper Kiiskinen Management portal for social mobilegame's administration 38 pages 23 May 2016
Degree	Bachelor of Engineering
Degree Programme	Information and Communciations Technology
Specialisation option	Software Engineering
Instructors	Antti Laiho, Senior Lecturer Ville Sillanpää, CTO

The purpose of this study was to develop a web application for Vulpine Games. The web app should be able to monitor and moderate a game world called Last Planets. Last Planets is a massive multiplayer strategy game for iOS, where all players are in the same universe and rule their own little planet.

The management portal was developed with Unity3D and C#. With Unity3D is possbile to develop software for different platforms including WebGL. The WebGL application can be used from a browser, which allows access to users from anywhere. Authentication was implemented by using the OAuth 2.0 protocol. At start up, the portal will get an access token from Azure Active Directory using the portal's client credentials. With the access, the token portal can use the backends API. With the API, the portal can communicate with the game world and backend. Portal was programmed to fetch the latest data from game world through the API and then to render that to the user. For the user to better understand the information, a few tools were made to the portal. One of them was a heatmap that helps analyzing the values of the game worlds on the map, and another one was piecharts that helps to understand the distribution of values.

From the portal the user can monitor the evolution of the gameworld and information of players and alliances as well as look at the issuetickets. The portal also gives the user a basic control over the gameworld. The user can for example give and remove bans from players and also to create new events to the game world. The portal gives a lot of extra value to the game development. Extra value for the portal is obtained by the fact that the portal allows easily visualization of a lot of different type of data in the game world. The portal also allows making heat maps that helps in the monitoring of the realm. With these tools the game world can be easily analyzed. In addition, the developers can make decisions more easily about what should be improved next, so that the game becomes an even better experience for the players.

In conclusion all the planned features were completed in the management portal in time. In addition to that some additional features were implemented in the portal such as pie charts. The portal's development and deployment will continue. After the soft-launch it will be easier to say what tools need to be implemented next to the portal, so that the management and designing of the game will become even easier with the portal.

Keywords	management portal, Unity3D, WebGL, C#, OAuth, data visualization

Sisällys

Lyhenteet

AoT	Ahead of Time compilation. Ohjelmointikielen kääntämistä binäärikoodiksi ennen sen suorittamista.
API	Application programmin interface eli ohjelmointirajapinta. Sen avulla eri ohjelmat voivat keskustella keskenään.
GUID	Globally Unique Indentifier. 32 heksaluvun uniikiksi tarkoitettu tunnusnumero, joka on ryhmiteltynä yhdysmerkein.
HTTPS	Hyper Text Transfer Protocol Secure. Hypertekstin siirtoprotokolla, jota selaimet ja www-palvelimet käyttävät tiedonsiirtoon.
JSON	Javascript object notation. Yksinkertainen tiedostomuoto tiedonvälitykseen.
Rest	Representational State Transfer. HTTP-protokollaan perustuva arkkitehtuurimalli ohjelmointirajapintojen toteuttamiseen.
Xml	Extensible Markup Language. Rakenteellinen kuvauskieli, jolla pystyy jäsentämään suuria tietomassoja selkeämmin.
WebGL	Web Grahics Library. Javacript kirjasto 2D- ja 3D-grafiikan renderöimiseen selaimessa ilman lisäosia.

1 Johdanto

Opinnäytetyön toimeksiantajana toimiva Vulpine Games on vuonna 2014 perustettu pelialan startup-yritys. Se tekee mobiilipelejä, jotka keskittyvät pelaajien sosiaaliseen kanssakäymiseen. Idea tähän opinnäytetyöhön tuli Vulpine Gamesilta. Vulpine Games on kehittämässä Last Planets -nimistä monin pelattavaa mobiilipeliä, johon se tarvitsee hallintaportaalin web-sovelluksena, jolloin siihen pääsee käsiksi mistä vain. Peli tulee olemaan free to play eli ilmaispeli. Peliä ei ole vielä julkaistu, vaan se on vielä aikaisessa kehitysvaiheessa. Pelin soft-launch eli sen julkaiseminen aluksi pienellä markkina-alueella, esimerkiksi vain Kanadassa, tapahtunee kevään 2016 aikana. Soft-launch on eräänlainen suljettu beta-testi. Peliä parannetaan tänä aikana seuraamalla, mitä pelissä tapahtuu, ja pelaajilta saadun palautteen perusteella. Soft-launchin jälkeen peli julkaistaan suuremmassa mittakaavassa kaikkien saataville. Siihen mennessä peli on kehittynyt paremmaksi elämykseksi, jolloin pelaajat viihtyvät sen parissa pidempään. Mobiilipeleissä juuri ensimmäinen pelikerta on tärkein, sillä keskimäärin 40 % pelaajista ei käynnistä ilmaispeliä enää toista kertaa [1].

Soft-launchiin mennessä hallintaportaalin pitäisi olla käyttökunnossa ja tuotannossa, jotta portaalin työkalut saadaan heti käyttöön auttamaan pelimaailman hallinnassa ja seuraamisessa. Tällöin itse pelinkehittämiseen jää enemmän aikaa, eikä tarvitse manuaalisesti yrittää seurata ja hallita pelimaailmaa. Hallintaportaalin tarkoituksena on siis pystyä hallitsemaan ja seuraamaan pelimaailman tapahtumia. Last Planetsin pelimaailma on dynaaminen, mikä tarkoittaa sitä, että maailma kasvaa pelaajien määrän lisääntyessä ja maailma myös muuttuu pelaajien pelatessa peliä. Pelissä on myös automaattisia tapahtumia, esimerkiksi maailmaan voi ilmestyä harvinainen reliikki. Nämä tapahtumat syntyvät automaattisesti tiettyjen algoritmien mukaan, joten portaalin pitää pystyä seuraamaan, miten maailman generointi toimii, suuressa ja pienessä mittakaavassa.

Hallintaportaalissa on myös erilaisia datanvisualisointimahdollisuuksia, kuten lämpökarttoja ja ympyräkaavioita. Niillä voidaan tarkemmin analysoida pelimaailmaa ja pelaajien käyttäytymistä. Näiden perusteella saatavasta datasta voidaan tehdä peliin tärkeitä korjauksia ja parannuksia.

Pelkkä pelimaailman seuraaminen ei kuitenkaan ole portaalin ainoa funktio, vaan portaalilla myös pystyy vaikuttamaan maailmaan ja moderoimaan sitä. Esimerkiksi portaalista pitää pystyä antamaan pelaajille porttikieltoja, luomaan uusia tapahtumia ja ratkaisemaan ongelmatikettejä.

2 Hallintaportaalit yleisesti

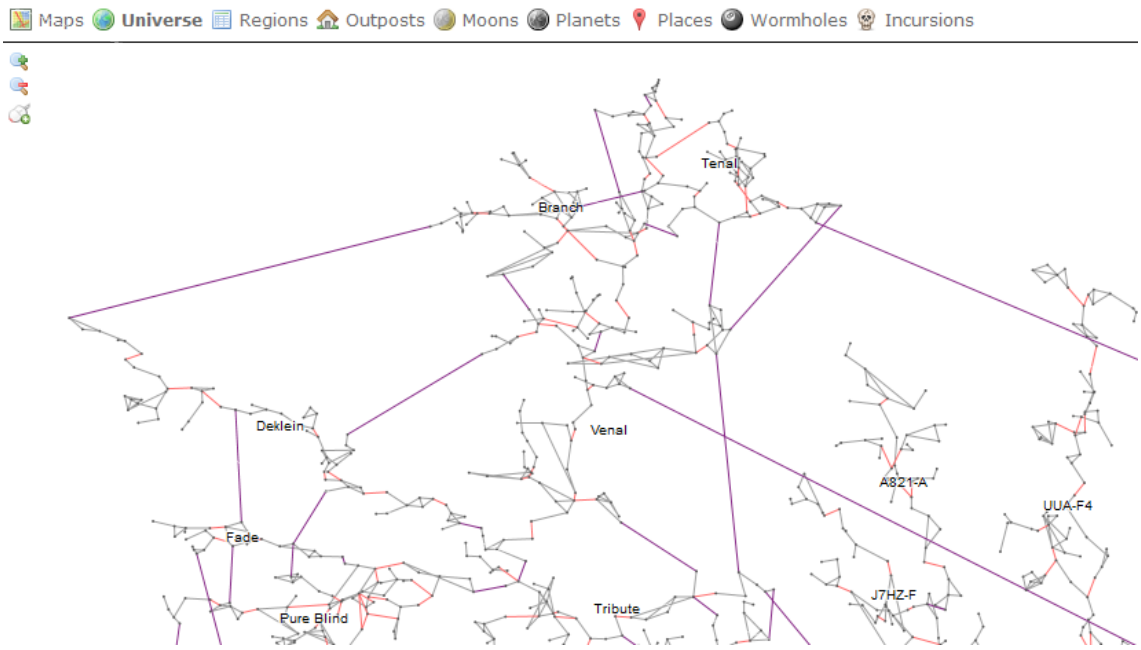
Hallintaportaali on lyhyesti esitettynä työkalu tai kokoelma erinäisiä työkaluja, joiden avulla hallinnoidaan tai jotka auttavat hallinnoimaan jotain tiettyä asiaa. Hallintaportaali voi olla joko nettisivu tai ohjelmisto. Hallintaportaaliin voidaan kuitenkin helposti lisätä uusia toiminnallisuuksia tai integroida uusia työkaluja sitä mukaa, kun sen vaatimukset muuttuvat. Hallintaportaalia voidaan tehdä samaan aikaan kun sovellusta kehitetään tai sen jälkeen kun sovellus on jo valmis. Hallintaportaaleissa on yleensä työkaluja, joilla helpotetaan itse kehitystyötä tai tarjotaan työkaluja sovelluksen ylläpitoon. Portaalia kannattaa siis kehittää jo samaan aikaan kun itse sovellusta ollaan kehittämässä. Hallintaportaalista yritys saa näin ollen taloudellista hyötyä, kun kehitystyö ja ylläpitovaihe helpottuvat.

Hallintaportaali on tärkeä työkalu nopeassa, ketterässä ja iteroivassa työskentelyssä. Sillä helpotetaan kehittäjien ja suunnittelijoiden arkea, kun osa toiminnallisuuksista voidaan automatisoida. Portaalissa on hankalien töiden tekemiseen tehty helpottavia työkaluja, ja erilliset työkalut saadaan tuotua portaalissa yhteen ja paremmin hahmotettavaan muotoon.

Tässä tapauksessa hallintaportaalilla hallitaan pelimaailma ja samalla seurataan pelimaailman tapahtumia, sen kehitystä ja pelaajien käyttäytymistä. Pelin moderointi helpottuu huomattavasti hallintaportaalin työkaluilla. Ilman hallintaportaalia pelimaailmaa voisi hallita vain kömpelösti tietokannasta tai pienillä erillisillä työkaluilla. Myös suuren pelimaailman seuraaminen olisi todella hankalaa, ilman sitä varten kehitettyjä työkaluja portaalissa.

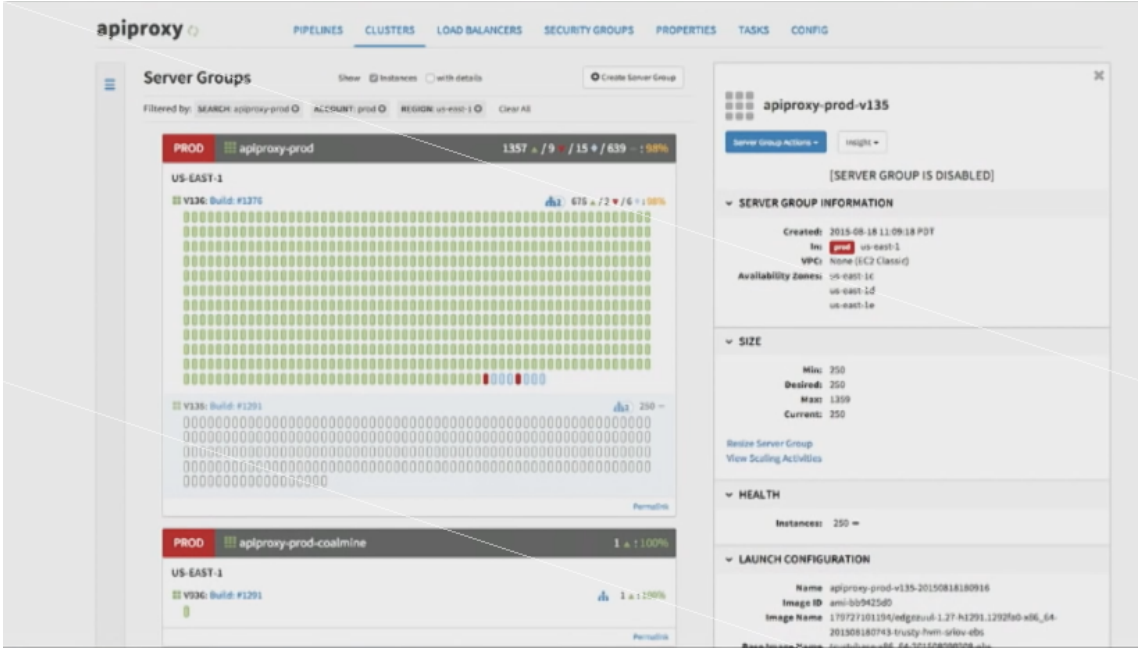
Kuvassa 1 on EVE Online -massiivimoninpelin maailma visualisoituna internetsivulla. Tämä ei ole pelinkehittäjien työkalu, vaan heidän APIn kautta tarjoamillaan ominaisuuksilla toteutettu maailman tilannekatsaus. Se on hyvin samantapainen kuin Last Planets -pelin portaalin halutaan olevan. Tämä visualisaatio auttaa huomattavasti

käyttäjiä hahmottamaan maailman koon ja havaitsemaan, mitä missäkin tapahtuu. Sen avulla käyttäjät voivat suunnitella hyökkäyksiä ja seurata, miten muut liittoumat muuttuvat ja leviävät maailmassa.



Kuva 1. Eve online-maailman visualisaatio, jonka peliyhteisö on tehnyt. Even tarjoamalla APIlla. [2.]

Kuvassa 2 on Spinnaker, jolla hallitaan pilvessä olevia sovelluksia ja infrastruktuuria. Netflix käyttää tätä web-portaalia hallitsemaan massiivista taustajärjestelmää. Spinnaker on uudempi version Netflixin vanhemmasta avoimen lähdekoodin työkalusta nimeltä Asgard. Netflixillä ei ole peliä, mutta heillä on miljoonia yhtäaikaisia käyttäjiä ja suuri taustajärjestelmä, jota pitää monitoroida ja ylläpitää jatkuvasti. Spinnaker auttaa Netflixin ylläpitämisessä ja monitoroinnissa. Siitä näkee helposti tärkeää informaatiota ilman, että pitäisi tietää, miten työkalua käytetään. Netflix on onnistunut automatisoimaan suuren osan infrastruktuuristaan, koska Netflixin työkalut ovat laadukkaita, ne parantavat tuottavuutta ja vähentävät kuluja. [3.]



Kuva 2. Spinnaker on Netflixin käyttämä hallintaportaali. Punaisella värillä näkyy heti pari instanssia, joissa on häiriöitä. [3.]

3 Portaalin teknologiset ratkaisut

Tässä kappaleessa käydään läpi portaalin erinäisiä teknologisia ratkaisuja. Kappaleessa tarkastellaan niiden historiaa sekä sitä, miten ne toimivat ja miksi niitä päätettiin käyttää. Lisäksi käydään läpi eteen tulleita ongelmia näiden teknologioiden kanssa ja kerrotaan, miten nämä ongelmat saatiin ratkaistua. Läpi käydään myös taustajärjestelmää ja sen APIa ja erilaisia rajapintoja, joita portaali käyttää hyväkseen.

3.1 HTTP ja JSON

HTTPS eli Hypertext Transfer Protocol Secure on protokolla, jota selaimet ja WWW-palvelimet käyttävät tiedonsiirtoon. Se on yhdistelmä HTTP- ja TLS/SSL-protokollia. Ensimmäinen HTTP:n versio julkaistiin vuonna 1991. HTTP-viestit koostuvat kyselystä asiakkaalta palvelimelle ja palvelimen vastaukselta asiakkaalle. [4.]

Vastausviestin rakenne koostuu vastauskoodista, kuten HTTP/1.1 200 OK, 404 Not Found ja niin edelleen. Vastauskoodi kertoo asiakkaalle, mitä kyselylle tapahtui. Tämän

jälkeen tulevat otsakekentät, joissa on yleistä tietoa viestistä. Ne koostuvat ”nimi:arvo”-pareista. Lopuksi otsakekenttien jälkeen tulee vaihtoehtoinen viestin sisältö.

Kysely koostuu myös header-kentistä ja mahdollisesta sisällöstä. Kysely kuitenkin alkaa jollain kyselymetodilla, kuten metodeilla GET, POST, PUT tai DELETE. Kyselymetodi kertoo palvelimelle, mitä asiakas haluaa tehdä. GET-metodi kertoo palvelimelle, että asiakas haluaa jotain dataa, kuten ladata HTML-sivun. POST-metodi taas kertoo, että asiakas haluaa lähettää palvelimelle jotain tietoa.

JSON eli JavaScript Object Notation on avoimen standardin kevyt datansiirtoformaatti. Sitä on ihmisen helppo lukea ja kirjoittaa, ja se on myös helppo jäsentää ja generoida tietokoneella. [5.] JSON koostuu kokoelmista nimi- ja arvopareja. JSON arvo voi olla objekti, taulukko, numero, merkkijono, tosi, epätosi tai null. [6.]

```
GET https://example.lp-portal.net/api/Galaxymap/Getallclusters HTTPS/1.1

HTTP/1.1 200 OK
DATE: Mon, 15 Feb 2016 22:06:27 GMT
SERVER: Microsoft-IIS/8.5
Last-Modified: Sun, 26 Sep 2010 22:04:35 GMT
CACHE-CONTROL: no-cache
PRAGMA: no-cache
Content-Length: 162
Connection: close
CONTENT-TYPE, application/json; charset=utf-8

[{"clusterId":"a8057a20-9d09-431f-cb6ee94a80a1dd5","positionX":0,"positionY":0},
{"clusterId":"098e3fad-b186-41ef-bf7fb99d31ec015f","positionX":0,"positionY":1}]
```

Kuva 3. Get Request ja HTTP-vastauksen headerit ja JSON-vastaus.

Kuten kuvasta 3 näkyy, tehdään GET-kysely taustajärjestelmän APIn, josta pyydetään kaikki pelimaailman klusterit. HTTP-vastauksen otsakkeiden jälkeen viestin sisällössä on JSON-formaatissa pyydetty data. Viestissä on JSON-taulukko GUID clusterId, int positionX ja int positionY. Tämä JSON-viesti saadaan jäsennettyä helposti Klusteri-luokaksi, kuten myöhemmin tullaan näkemään. Kaikki data portaalin ja APIn välillä kulkee JSON-formaatissa HTTP-viestien avulla.

JSON Serialize/Deserialize -kirjastona käytetään avoimen lähdekoodin FullSerializer-kirjastoa, koska se toimii hyvin kaikkien alustojen päällä, joita Unity tukee mukaan lukien WebGL. FullSerializeria voi myös ajaa merkkikokoriippumattomana, mistä on paljon hyötyä, sillä APIn JSON-vastauksissa avaimet ovat pienellä alkukirjaimella

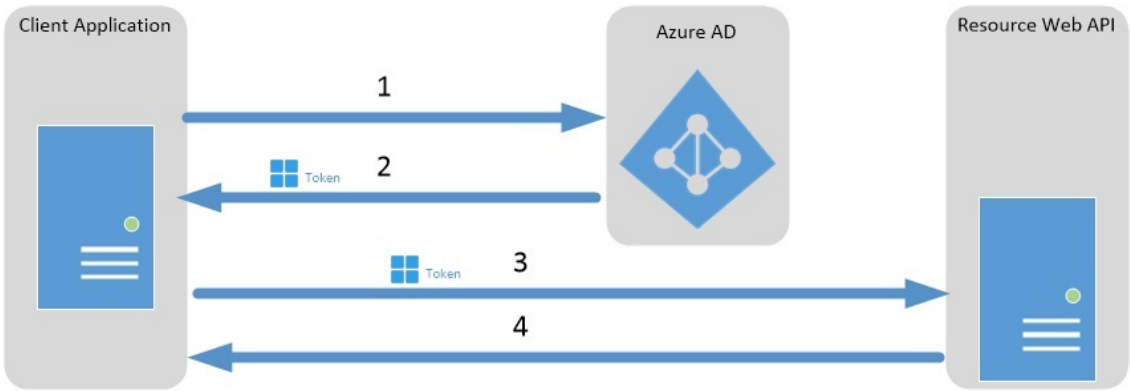
kirjoitettuja, mutta luokissa, joihin arvot jäsennetään, muuttujat ovat isolla alkukirjaimella kirjoitettuja. [7.]

3.2 OAuth 2.0

OAuth 2.0 on avoimen standardin autentikaatioprotokolla. Yleisesti OAuth tarjoaa käyttäjälle turvallisen pääsyn taustajärjestelmän resursseihin. Alkuperäinen OAuth julkaistiin vuonna 2006. Vuonna 2012 julkaistiin OAuth 2.0 standardi. [8.]

Portaalissa käytetään client credientals autentikaatiomenetelmää. Se on service to service -metodi. Tätä käytetään silloin, kun asiakas on luottamuksellinen. Portaali siis käyttää omia tunnuksiaan autentikoitumaan eikä esiinny käyttäjänä. [9.]

Kuvasta 4 nähdään, miten client credietals -menetelmällä pyydetään access tokenia eli eräänlaista autentikoitumisavainta Azure autentikaatiolta. Jos pyyntö on validi, niin vastaukseksi saadaan JSON-objekti, jossa on mukana access token. Tämän jälkeen se lisätään HTTP-pyyntöihin, joilla voidaan autentikoitua palvelimelle ja päästä suojattuihin resursseihin käsiksi. Access tokeneilla voi olla erilaisia oikeuksia, kuten vain lukuoikeus tietoihin. [10.]



Kuva 4. Client credientals -autentikaatiokaavio Azure ActiveDirectoryn ja palvelimen kanssa. [10.]

Autentikointi tapahtuu kuvan 4 mukaisen kaavion logiikalla. Kohdassa 1 lähetetään HTTP-viesti. Tämän viestin sisällön mukana täytyy olla ”grant_type: <client_credientals>”, ”client_id: <portaalin id>” ja ”client_secret <salainen avain>”. Lopuksi viestissä on oltava ”resource: <mihin resursseihin ollaan haluamassa

oikeuksia>”. Jos pyyntö on validi eli portaalin id ja asiakkaan salainen avain täsmäävät, niin Azure AD generoi uuden access tokenin, jolla pääsee käsiksi palvelimen resursseihin, joka vastaa takaisin HTTP-viestillä, jossa on tämä access token.

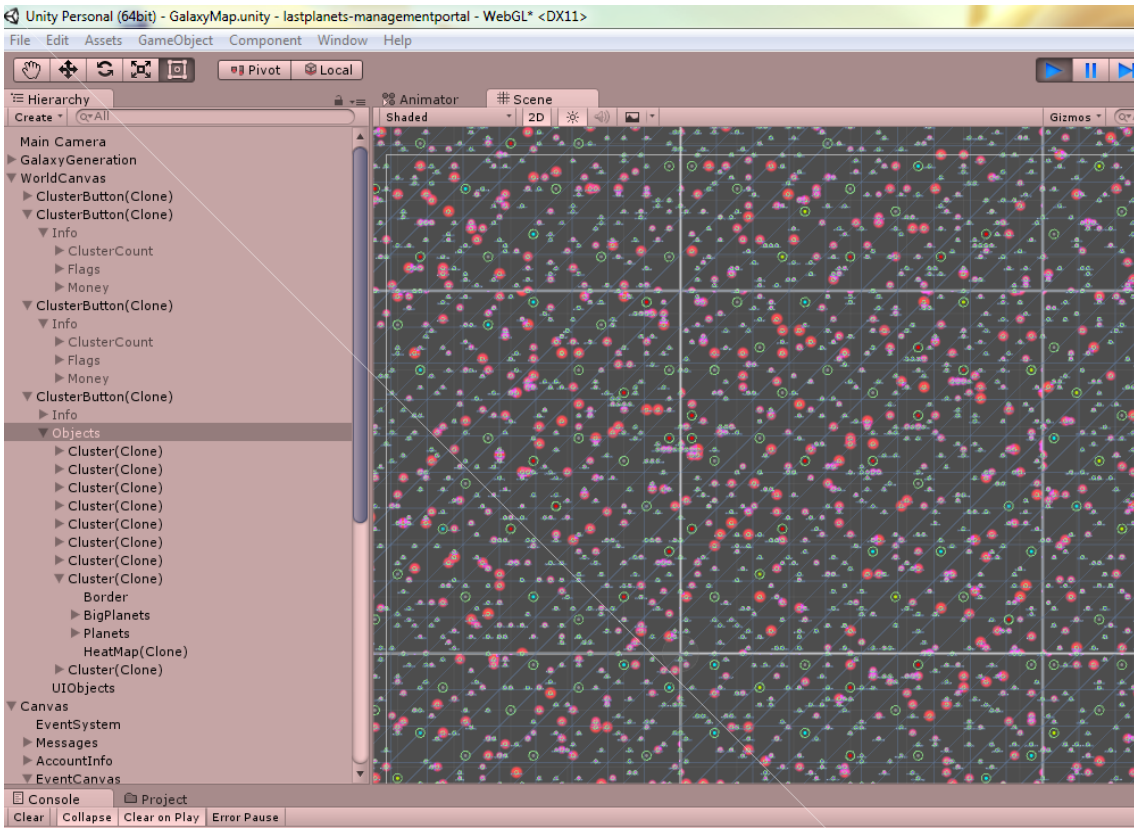

```
{
"access_token":"eyJhbGciOiJSUzI1NiIsIng1dCI6IjdkRC1nZWNOZ1gxWmY3R0xrT3ZwT0IyZGNW
QSIsInR5cCI6IkpXVCJ9.eyJhdWQiOiJodHRwczovL3NlcnZpY2UuY29udG9zby5jb20vIiwiaXNzIjo
iaHR0cHM6Ly9zdHMud2luZG93cy5uZXQvN2Z1ODE0NDctZGE1Ny00Mzg1LWJlY2ItNmRlNTdmMjE0Nzd
lLyIsImlhdCI6MTM4ODQ0ODI2NywibmJmIjoxMzg4NDQ4MjY3LCJleHAiOjEzODg0NTIxNjcsIn...",
"token_type":"Bearer",
"expires_in":"3599",
"expires_on":"1388452167",
"resource":"https://service.example.com/"
}
```


Kuva 5. Esimerkki vastausviestistä, jonka AzureAD palauttaa, jos autentikaatio on onnistunut.

Onnistunut vastaus näyttää kuvassa 5 näkyvältä JSON-objektilta. Vastauksessa saadaan access token, jolla voidaan autentikoitua nyt taustajärjestelmän APIlle. Access tokenia käytetään lisäämällä se HTTP-kyselyn otsakkeisiin, jossa avain on ”Authentication” ja arvoksi tulee Bearer <access token>. Vastauksena saadaan myös expires_in-arvo, jonka jälkeen access token vanhentuu. Tästä arvosta voidaan laskea, milloin access token täytyy pyytää uudelleen, jos portaalia ollaan vielä käyttämässä. [11.]

3.3 Unity3D ja WebGL

Last Planets on tehty Unity3D:llä, joten on järkevää käyttää sitä myös portaalin tekemiseen. Näin ollen pelin asiakaspuolen toteutuksia voidaan käyttää portaalissa. Esimerkiksi taisteluiden uusintojen katsomisen saisi portaaliin helposti integroitua omana työkalunaan, eikä sitä tarvitsisi tehdä useampaan kertaan. Taustajärjestelmä ja peli käyttävät kumpikin jaettua C#-kirjastoa, jossa on määriteltynä yhteisiä luokkia. Tämä jaettu kirjasto sisältää useita pelin kannalta tärkeitä toimintoja ja logiikkaa, kuten maailmangenerointialgoritmin. Kyseinen Common-kirjasto on lisätty jokaisen projektin git-version hallintaan submoduuliksi, jolloin sitä on helppo käyttää. [12.] Tästä syystä myös päädyttiin käyttämään Unityä, koska se tukee C#-kieltä, jolloin voidaan käyttää projektin yhteisiä kirjastoja helposti.



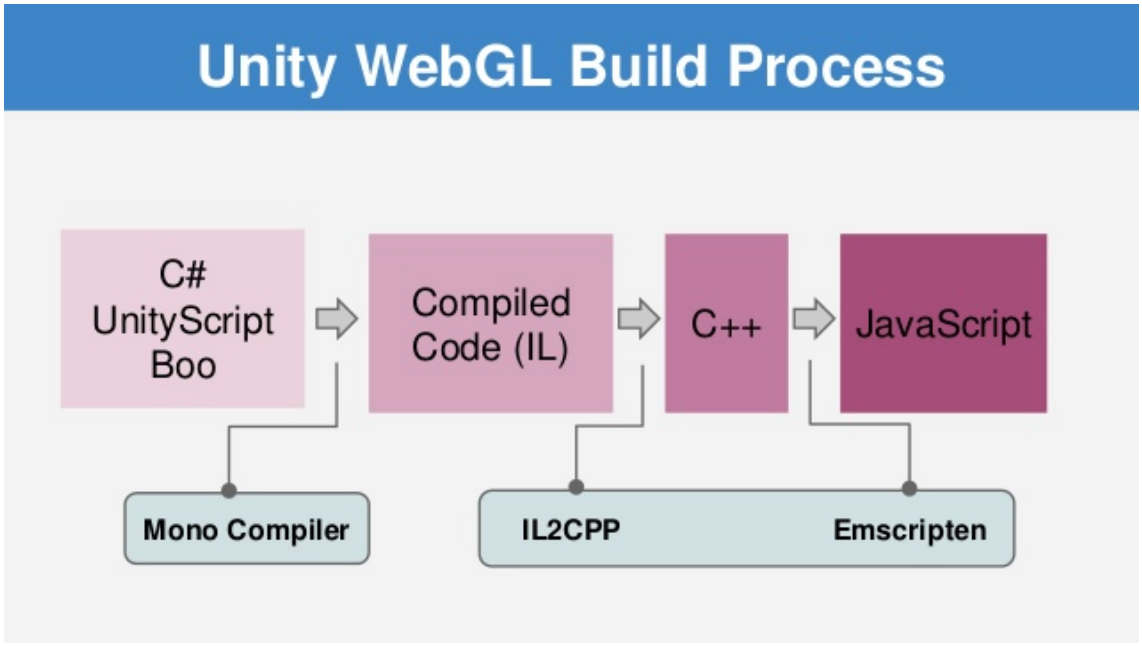
Kuva 6. Unity editor. Vasemmalla näkyy scenen hierarkia ja oikealla scene, jossa on valittuna planeetat ja lämpökartta pohjalla.

Unity3D on Unity Technologiesin kehittämä pelimoottori, joka tukee useita eri alustoja. Se on tämän hetken suosituin pelimoottori [13]. Unity3D tarjoaa käyttäjälle monia helppokäyttöisiä työkaluja nopeaan sovelluskehitykseen. Kuvassa 6 näkyy Unity editor, jolla sovelluksia kehitetään. Unityllä voi ohjelmoida kolmella eri kielellä. Näitä ovat Javascript, C# ja Boo. C#, jota tässä projektissa käytetään, on Microsoftin kehittämä ohjelmointikieli, joka julkaistiin kesäkuussa 2000. C#:n tarkoitus on yhdistää C++:n tehokkuus ja Javan helppokäyttöisyys. C#-syntaksi on lähellä C-kielen syntaksia. [14.]

Unity on juuri tiputtanut vanhan web-tukensa pois uusimmasta versiota, mutta uusi WebGL-tuki on tullut jokin aika sitten Unityyn, joten tultiin siihen tulokseen että sitä kannatta käyttää. Uusi WebGL-tuki on Unityssä vielä eräänlaisessa beta-vaiheessa, joten haasteita saattaa ilmaantua.

WebGL on Javascript API 3D- ja 2D-grafiikan renderöimiseen yhteensopivissa internetselaimissa ilman ylimääräisiä liitännäisiä. Vuonna 2009 yleishyödyllinen yhteisö

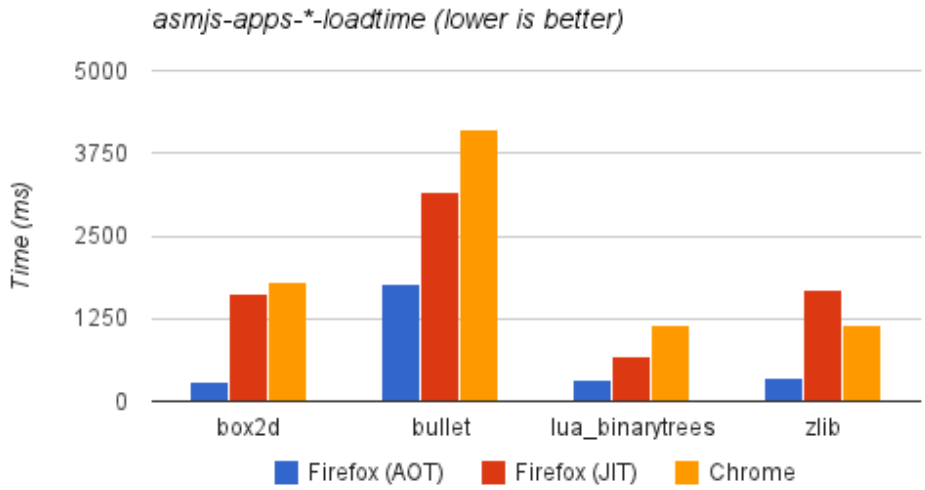
Khronos Group aloitti WebGL:n kehittämisen. Maaliskuussa vuonna 2011 julkaistiin WebGL:n 1.0 spesifikaatio. [15.]



Kuva 7. Unity3D:n prosessi kääntää koodin WebGL-ohjelmaksi [16].

Unity3D:llä voidaan siis rakentaa WebGL-ohjelmia, jotka toimivat selaimessa. Kuten kuvasta 7 nähdään, prosessi on monimutkainen ja se toimii niin, että Unity kääntää C#-koodin ja Unity-pelimoottorin koodin monokääntäjän avulla. Mono on avoimen lähdekoodin .NET-kääntäjä, joka toimii kaikilla alustoilla. [17;18]

Tämän jälkeen käännetty koodi käännetään IL2CPP-koodiksi, joka on paljon nopeampaa kuin C#-koodi. IL2CPP:ssä ei ole roskienkeruuta, joka hidastaisi koodia. [19.] IL2CPP-koodi voi olla 2-3 kertaa nopeampaa kuin pelkkä C#-skripti Unityssä. IL2CPP:stä saadaan C++-koodia. C++-koodi käännetään vielä tämän jälkeen Emscriptenin avulla Javascriptiksi [20.] Emscripten on kääntäjä, joka kääntää C/C++-kieliä optimoiduksi Javascript-koodiksi asm.js-formaattiin. [21.] Asm.js on erittäin rajoitettu osajoukko Javascriptiä, joka on optimoitu suorituskykyä varten. Se kuitenkin käyttäytyy identtisesti javacriptiin verrattuna. [22.] Jos selain, jolla WebGL-sovellusta suoritetaan, tukee AoT (Ahead of Time)-Javascript-kääntäjää, niin asm.js voidaan suorittaa vielä nopeammin kuin normaalilla Javascript-kääntäjällä, kuten kuvasta 8 voidaan havaita.



Kuva 8. Asm.js-koodin suorituskykytesti Firefox-selaimella AoT- ja JIT-kääntäjillä sekä Chrome-selaimella. [23.]

Firefox- ja Edge-selaimet ajavat WebGL:ää tällä hetkellä parhaiten, koska ne käyttävät optimoitua AoT-kääntäjää kääntäessään asm.js-koodia binäärikoodiksi. Kuitenkin lopullinen käännös natiivikoodista asm.js-koodiksi suoriutui melkein kaksi kertaa hitaammin kuin natiivikoodi. [24.] Grafiikan piirto on kuitenkin lähellä natiivisovellusten suoritusta, sillä WebGL osaa käyttää hyväkseen näytönohjainten rautakiihdytystä.

Lopullinen sovelluksen käännös Javascriptiksi tarkoittaa siis sitä, että kaikkia C#:n ominaisuuksia ei voida käyttää, sillä lopullinen WebGL-paketti ei tue kaikkia ominaisuuksia, joita C# tukee. [25.] Rajoituksia verrattuna C#-kieleen ovat esimerkiksi threadit, ääniominaisuudet, skriptien testaus ja networking. Nämä rajoitukset myös hidastavat hieman jotain osia WebGL-sovelluksissa, sillä jotkin osa-alueet Unity3D-moottorissa on optimoitu käyttäen threadeja, ja niitä ei Javascript vielä tue.

Kuten aikaisemmin mainittiin kohdassa 3.1, JSON Serialize/Deserialize -kirjastoksi valittiiin FullSerializer. Suurin syy tähän on, että se ei käytä .NETin reflection-ominaisuutta, joka ei toimi Javascriptissä. Todella moni C#-kielen JSON Serialize/Deserialize kirjasto käyttää tätä ominaisuutta, ja se ei tässä projektissa kelpaa. Koska tämä kirjasto on myös avoimen lähdekoodin projekti, pystyttiin siihen tekemään muutoksia. Yksi muutos tehtiin DateTime-luokkien jäsentämiseen. FullSerializer ei osannut jäsentää formaatissa 2016-01-22T12:06:57.503005Z olevaa päivämäärää ja se korjattiin FullSerializer kirjastoon.

Submoduulina olevassa Common repositoryssä oli myös koodia, joka ei toiminut WebGL-paketissa. Näitä osia Common-kirjastosta jouduttiin korjaamaan. Osa korjauksista oli helppo toteuttaa, kuten kuvassa 9 näkyvä muutos. Siinä pystyttiin käyttämään vaihtoehtoista toteutusta, mikä kuitenkin antaa saman lopputuloksen kuin alkuperäinen toteutus.

```
-            if (channelDataFromMessage.IsNullOrEmpty() == false)
+            if (String.IsNullOrEmpty(channelDataFromMessage) == false)
```

Kuva 9. Muutos yhteiseen koodipohjaan, jotta WebGL käännös menee läpi.

Osassa koodia jouduttiin kuitenkin käyttämään alustaspesifistä koodia, kuten kuvasta 10 havaitaan. Ylempi osa koodista suoritetaan, kun alustana on jokin muu kuin WebGL, ja #else-merkin jälkeen oleva osa ajetaan vain WebGL-projekteissa:

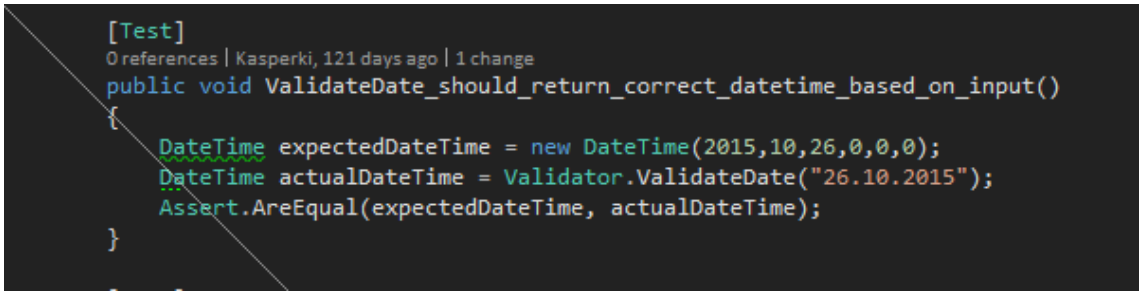
```
+#if !UNITY_WEBGL
             StackFrame frame = new StackFrame(1, false);
             var method = frame.GetMethod();
             var type = method.DeclaringType;

             return GetLogger(type.FullName);
+#else
+            return GetLogger("SubServerCommon.Utilities.ConfigBase");
+#endif
```

Kuva 10. Hieman isompi muutos koodipohjaan, jossa on alustaspesifistä koodia

Hallintaportaaliprojektin kylkeen on myös kirjoitettu pieni määrä yksikkötestejä käyttäen NUnit frameworkia. NUnit on yksikkötestaus-framework kaikille .NET-kielille. NUnitia on alun perin lähdetty kehittämään JUnitin pohjalta. [26.] Testejä on kirjoitettu vain portaalin omille luokille, jotka eivät periydy Unityn MonoBehaviour luokasta, koska niiden testaaminen ilman Unityn omia testityökaluja on hankalaa.

Koodista testattu on lähinnä apuluokkia, kuten validointiluokkaa, joka validoi käyttäjän antaman syötteen ja tarkistaa, onko käyttäjän antamalla syötteellä olevia asioita olemassa, jotta niitä voidaan lähettää APIlle eteenpäin. Kuvassa 11 testataan metodia, jonka tehtävä on validoida ja jäsentää käyttäjän syöte ja katsoa, voiko annetusta

merkkijonosta luoda päivämäärää ja onko päivämäärä sallittu. Tämä tarkoittaa sitä, että päivämäärän pitää olla tulevaisuudessa.



Kuva 11. Yksikkötesti, jossa validaattori tarkistaa, onko käyttäjän syöttämä aika validi.

Lopullinen WebGL-koontiversio sisältää muutamia tiedostoja, kuten *Index.html* -tiedoston, jonka avulla selain osaa ladata oikean sisällön käyttäjälle. *Projekti.js* -tiedosto sisältää projektin lopullisen käännöksen ohjelmakoodin. *Projekti.data* -tiedostossa on Unityn scene -tiedostot ja projektin kuvat, äänet, fontit ja niin edelleen. *Projekti.js.mem* -tiedosto initialisoi ajonaikaisen kekomuistin ja *UnityLoader.js* lataa Unityn sisällön nettisivulle. [27.] Nämä tiedostot laitetaan pyörimään jonkin palvelinohjelman, esimerkiksi Apachen päälle. Apache on HTTP-palvelin, joka on avoimen lähdekoodin projekti. HTTP-palvelin jakaa tiedostoja HTTP-protokollan yli, jolloin tiedostoihin pääsee käsiksi internetissä. Näin hallintaportaali saadaan käyttäjien saataville. [28.]

3.4 Taustajärjestelmän API

Taustajärjestelmä sijaitsee Azuren pilvipalvelussa, ja se ajaa redis-tietopalvelinta, joka on suosituin key-value NoSQL-tietokanta. [29.] Tietokantana taustajärjestelmässä käytetään Azure Storagea, joka replikoituu jatkuvasti toisessa datakeskuksessa. Sieltä voidaan lukea ilman, että pelin käyttämä tietokanta joutuu stressin alle. Redistä taas käytetään nopeasti vaihtuvalle väliaikaiselle datalle.

NoSQL:n etuna perinteiseen SQL-tietokantaan on sen skaalautuvuus helposti pilvessä, ja se, että se pystyy suoriutumaan hyvinkin suuresta samanaikaisesta määrästä luku- ja kirjoitusoperaatioita verrattuna normaalin SQL-relaatiotietokantaan. [30.] NoSQL-tietokannat ovat tyypillisesti kuitenkin tinkineet joistain relaatiokantojen ominaisuuksista, kuten eheys- tai turvallisuusominaisuuksista.

Palvelimella pelin tietokanta pitäisi kopioida esimerkiksi kerran päivässä toiseen tietokantaan. Tästä tietokannan kopiosta voitaisiin louhia ja laskea mielenkiintoista dataa, joka auttaisi pelinkehittämisessä. Hallintaportaali voisi myös tehdä useita kyselyitä tähän uuteen kantaan käyttäen APIa, eivätkä ne vaikuttaisi peliin ollenkaan. Kuitenkin lähtevien HTTP-kyselyiden täytyy mennä APIn kontrollereiden kautta pelin kantaan, jolloin muutokset peliin tulevat voimaan peliin, eivätkä mene kannan kopioon.

API jota portaali käyttää, on koodattu C#:lla, ja se käyttää samaa common submoduulia kuin muutkin projektit. API perustuu REST (*Representational State Transfer)*-arkkitehtuurimalliin. REST-rajapinta perustuu HTTP-protokollaan ja se tarjoaa käyttäjälle erilaisia päätepisteitä, jotka perustuvat objekteihin. Ohjelmointirajapinnan avulla ohjelmat voivat keskustella keskenään. Hyvä API tarjoaa käyttäjälle mustan laatikon, jolloin APIn käyttäjän ei tarvitse tietää, eikä välittää, miten API toimii pinnan alla. Vaikka APIin tehtäisiin muutoksia ja parannuksia, eivät sitä käyttävät ohjelmat häiriintyisi mitenkään, koska APIn tarjoama kerros pysyy aina samana.

Yksi esimerkki APIn tarjoamasta päätepisteestä on POST api/Events, jolla voidaan manuaalisesti tehdä uusia tapahtumia pelimaailmaan. Tämän päätepisteen dokumentaatio voidaan nähdä kuvasta 12. Tästä dokumentaatiosta nähdään, mitä tämä päätepiste haluaa ottaa vastaan ja mitä se tekee. Dokumentaatiosta voidaan myös nähdä, mitä päätepisteet vastaavat ja missä muodossa. Päätepisteet voivat olla avoimia kaikille, jolloin esimerkiksi pelin yhteisö voisi tehdä omia toteutuksiaan maailman tilanteesta, kuten tarjota työkaluja hyökkäysten suunnittelua ja liittoutumien laajentumista varten. Osa päätepisteistä voi taas vaatia autentikoinnin kuvan 3 vaatimalla tavalla. Tällä hetkellä yksikään päätepiste ei ole auki yhteisölle, vaan ne vaativat autentikoinnin. Portaali saa kaiken datansa pelimaailmasta tämän APIn avulla. [31.]

Body Parameters

EventStartData

Name	Description	Type	Additional information
EventType		EventType	Required
LayoutNumber		integer	Required
StartTime		date	Required
ClusterId		globally unique identifier	Required
AreaId		integer	Required
Name		string	None.

Request Formats

application/json, text/json

Sample:

```
{
  "eventType": 0,
  "layoutNumber": 1,
  "startTime": "2016-03-02T17:34:06.1687794+00:00",
  "clusterId": "a895ee77-b10b-490d-b673-919487e6a958",
  "areaId": 1,
  "name": "sample string 1"
}
```

Kuva 12. APIn päätepisteen dokumentointi. Pyytämällä päätepisteen data voidaan luoda uusia tapahtumia pelimaailmaan.

4 Hallintaportaalin vaatimukset ja toteutusprosessi

Tässä kappaleessa käydään läpi portaalin vaatimuksia ja suunnitteluprosessia. Kappaleessa kerrotaan, miten pelimaailman vaatimukset ja haasteet saatiin toteutettua ja mitä portaaliin on tehty, jotta se voi auttaa suunnittelijoita tekemään parempia päätöksiä, kun peliin pitää tehdä muutoksia.

4.1 Pelimaailman vaatimukset

Yksi portaalin vaatimuksista on, että sillä voidaan seurata ja selata koko pelimaailmaa. Se tarkoittaa sitä, että parhaimmillaan pelissä voi olla miljoonia pelaajia, joten tämä täytyy ottaa huomioon suunniteltaessa portaalia. Pelimaailma koostuu klustereista eli galaksijoukoista. Yhdessä klusterissa voi olla lähemmäs 1000 pientä planeettaa, jotka ovat automaattisesti tekoälyvastustajien hallinnassa. Tämän lisäksi yhteen klusteriin mahtuu myös 600 pelaajaa. Uuden pelaajan liittyessä peliin hän saa yhden tekoälyn hallinnassa olevan planeetan itselleen. Klustereissa on myös pieni joukko suurempia planeettoja, jotka ovat pelaajien kiinnostuksenkohteita. Niissä voi tällä hetkellä olla liittoutumia, aarteita, tapahtumia tai portaaleita.

Pelimaailma kasvaa pelaajien lisääntyessä. Kun peliin tulee tarpeeksi uusia pelaajia, niin taustajärjestelmä luo peliin sitä mukaan uusia klustereita. Klustereita myös liikutetaan automaattisesti maailmassa. Klusterit, joissa on eniten toimintaa, ovat maailman keskellä, ja ne klusterit, joissa ei ole enää aktiivisia pelaajia kovinkaan montaa, siirtyvät maailman laidalle. Tätä automaattista klustereiden siirtymistä pitäisi pystyä monitoroimaan portaalissa.

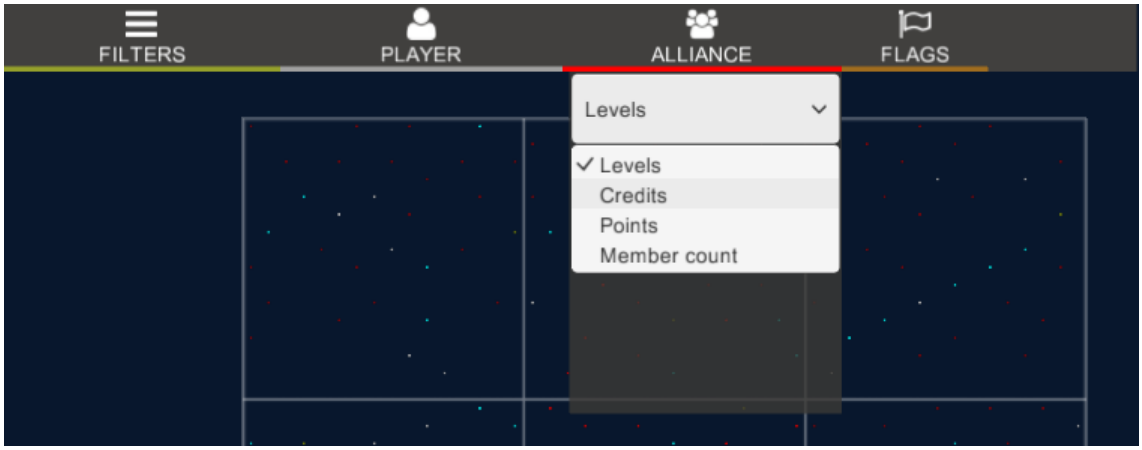
Pelimaailmaa ei ole myöskään jaettu erillisiin universumeihin, joten kaikki pelaajat ovat samassa maailmassa, jolloin pelimaailma vain kasvaa pelin vanhentuessa. Pelimaailma voi siis muuttua paljonkin hyvin lyhyessä ajassa sen lisäksi että pelaajia tulee lisää, sillä pelaajat voivat luoda suuremmille planeetoille liittoutumia ja rakentaa portaaleja, sekä myös vaihtaa paikkaa maailmassa. Tämän lisäksi tapahtumia myös generoituu automaattisesti pelimaailmaan. Näitä muutoksia halutaan seurata, jotta voidaan tarjota tämän tiedon pohjalta pelaajille parempia päivityksiä.

Portaalin testaaminen kehitysvaiheessa on ollut haastavaa, koska kehityspalvelimella on vain pari klusteria ja todella vähän pelaajia, joten suurempaa rasitustestausta ei voida suorittaa. Näin ollen portaalissa on oma testidatan generaattori, jolle pystyy antamaan parametreja eli kertomaan, kuinka paljon dataa halutaan generoida maailmaan. Data generoidaan suoraan välimuistiin, jolloin käytettäessä portaalia testidatan kanssa data löytyy ja haetaan aina välimuistista, eikä portaalin toteutusta tarvitse muuttaa. Ongelmana testidatan kanssa on, ettei se muutu, eikä se kuvasta maailman muutosta. Testidata kuvastaa vain tämänhetkistä tilaa, eikä siinä ole otettu huomioon kaikkia mahdollisuuksia, joita tuotantodatassa on mahdollista käydä pelaajien toimesta.

4.2 Käyttöliittymä

Käytön pitää olla mahdollisimman helppoa ja intuitiivista. Portaalin käyttö tällä hetkellä tapahtuu pelkällä hiirellä. Pikanäppäimiä on mahdollista lisätä, kun tiedetään, mitkä toiminnallisuudet ovat eniten käytettyjä. Tämä saadaan selville pidempiaikaisella testaamisella. Klusteriryhmien ja klustereiden näkymien vaihtaminen tapahtuu vierittämällä hiiren rullaa. Tämä on hyvin luonnollinen tapa: siinä mennään ylemmästä yleiskatsausnäkymästä tarkempaan näkymään lähemmäs planeettoja, joten sitä ei tarvitse opettaa käyttäjälle.

Hallintaportaalin yleiskatsausnäkymässä, jossa on listattuna klusteriryhmät, ei tällä hetkellä ole vielä mitään toiminnallisuutta, joten siinä ei ole myöskään muuta käyttöliittymää, kuin klusteriryhmiin siirtyminen. Kun siirrytään klusteriryhmään, avautuu käyttäjälle yläpalkki, jossa on erilaisia toimintoja. Yläpalkissa on useita kategorioita. Jokaisen kategorian alle on lajiteltu niille omat toiminnot. Kuten kuvasta 13 näkyy, käyttäjä on avannut Alliance-paneelin. Sen alta voidaan luoda lämpökarttoja tämän klusteriryhmän liittoutumiin liittyvistä tiedoista. Paneelien alla on hyvin tilaa lisätä sinne uusia toimintoja sitä mukaa kun tarvetta tulee. Ensimmäisestä Filter-paneelista voidaan vaikuttaa, siihen mitä klusterissa näytetään. On siis mahdollista piilottaa planeetat niiden tyyppien mukaan, jos halutaan tarkkailla vain joitain tiettyjä tietoja. Players näyttää pelaajiin liittyviä tietoja ja Flags hakee kartalle kaikki klusteriryhmän ongelmatiketit. Yläpalkin kuvakkeet ja muut kuvakkeet ovat Font Awesome -projektista, joka on avoimen lähdekoodin kuvakeprojekti. [32.] Kuvakkeet helpottavat käyttöä ja tekevät portaalista miellyttävämmän käyttää.



Kuva 13. Paneelit ryhmiteltynä objekteihin, joissa on niille tyypillisiä komentoja

Näkymät ovat ikkunoita, joista nähdään objektien tietoja tarkemmin kuin kartalta, ja niissä voidaan myös muokata joitain objektien tietoja. Näkymiin päästään käsiksi, kun mennään jonkin planeetan päälle hiirellä. Tämä antaa käyttäjälle erilaisia vaihtoehtoja riippuen, minkä planeetan päällä kursori on ja missä tilassa planeetta on tällä hetkellä. Vaihtoehtoja voidaan helposti lisätä sen mukaan, minkä planeetan päällä ollaan ja minkälaisia toimintoja sille voidaan antaa. Esimerkiksi Event-planeetan päällä ollessa saadaan plus-merkki, jota voidaan luoda planeetalle uusi tapahtuma manuaalisesti tai listan-merkki, josta painamalla nähdään, onko tapahtumaa menossa tai mikä tapahtuma planeetalla kenties on. Näkymistä kerrotaan tarkemmin kappaleessa 4.5.1.

4.3 Autentikointi

Kuvassa 15 on kuvattuna portaalin toimintalogiikka yksinkertaisesti. Kun portaali avataan selaimessa, se autentikoituu heti alkuunsa OAuth 2.0 ClientCredientals -systeemillä kappaleessa 3.2 kuvatulla tavalla. Tästä saatu access token pistetään muistiin, ja portaali käyttää sitä Request-apuluokan avulla liittämällä sen automaattisesti lähteviin HTTP-kyselyihin, jolloin päästään APIin kiinni. Access token haetaan uudestaan, kun se on vanhentumassa ja jos portaalia ollaan vielä käyttämässä.

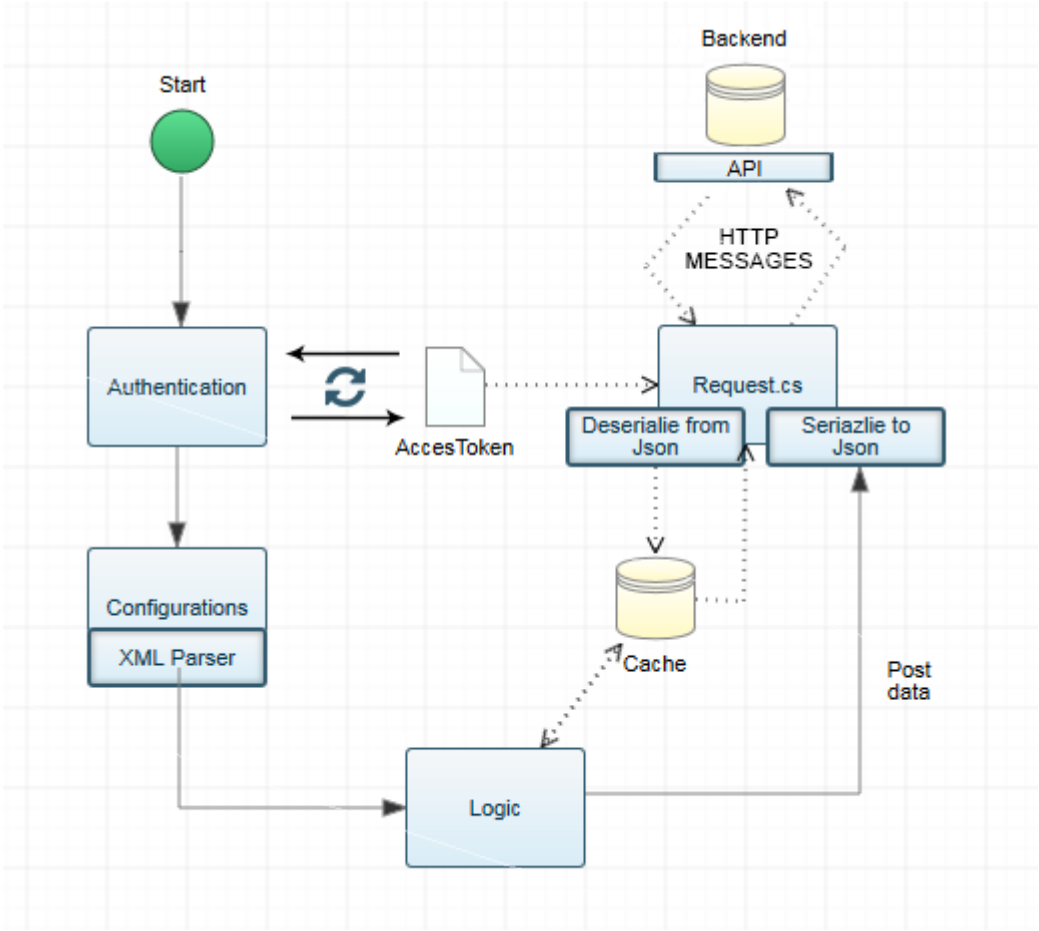
Onnistuneen autentikoinnin jälkeen portaali käy hakemassa pelin konfiguraatiotiedoston common submoduulin SHA-tiivistealgoritmin osoittamasta

paikasta. Se haetaan joka kerta uudestaan, kun portaali käynnistetään. Konfiguraatiotiedosto on pakattu gzip-tiedostoksi, sillä se pienentää ladattavan konfiguraation kokoa huomattavasti. gzip-tiedosto puretaan, ja puretusta tiedostosta saadaan konfiguraatio XML-formaatissa ja se parsitaan omiin luokkiinsa. Konfiguraatiotiedoston avulla peliä voi hienosäätää ilman, että peliä tarvitsisi päivittää. Kuvassa 14 on osa konfiguraatiotiedostoa. Siinä on määriteltynä maailman generointialgoritmin arvoja, kuten se, kuinka monta planeettaa yhteen klusteriin luodaan. Tiedostossa on määriteltynä paljon erilaisia asioita, esimerkiksi se, minkälaisia tapahtumia voidaan luoda. Hallintaportaali käyttää näitä luokkia, joihin konfiguraatio jäsennettiin apuna, ja näin ollen portaalia ei tarvitse päivittää, jos peliin tulee uusia tapahtumatyyppejä, vaan portaali osaa hakea ne dynaamisesti ja rakentaa näkymän niistä dynaamisesti käyttäjälle.

```
<GalaxyMapConstant>
  <PlayersInCluster>600</PlayersInCluster>
  <BotsInCluster>1800</BotsInCluster>
  <AllianceCount>20</AllianceCount>
  <EmptyCount>5</EmptyCount>
  <EventCount>5</EventCount>
  <RelayCount>1</RelayCount>
  <PlanetsInArea>4</PlanetsInArea>
  <AreaSize_SlotUnit>4</AreaSize_SlotUnit>
  <ClusterSize_SlotUnit>64</ClusterSize_SlotUnit>
  <GridSize_UnityUnit>9.0</GridSize_UnityUnit>
  <PaddingSize_UnityUnit>1.2</PaddingSize_UnityUnit>
  <ScoutExpirationMinutes>60</ScoutExpirationMinutes>
</GalaxyMapConstant>
```

Kuva 14. Osa pelin konfiguraatiotiedostosta XML-formaatissa.

Kun logiikassa tarvitaan jotain dataa, kuten pelaajan tiedot johonkin näkymään, kysytään välimuistista kuvan 15 tavalla haluttua dataa. Jos tieto löytyy välimuistista, palautetaan se käyttäjälle. Jos sitä taas ei löydy, luodaan HTTP-kysely APIlle ja haluttu tieto saadaan APIlta, mutta pienellä viiveellä. Logiikassa tämän takia kaikki metodit, joissa pyydetään dataa, toimivat asynchronisesti eli ne suoritetaan callbackin-toiminnon kautta. Ne siis kutsutaan silloin kun data on saatavilla.



Kuva 15. Kaavio logiikasta.

Kaikki APIn HTTP-kyselyiden JSON-vastaukset jäsennetään. Luvussa 3.1 esitelty FullSerializer hoitaa Request-luokassa tämän käännöksen JSONsta luokiksi ja toisinpäin. Vastaanotettu data tallennetaan välimuistiin. Näin saadaan vähennettyä turhaa liikennettä, kun samaa dataa haetaan uudestaan, sekä data saadaan haettua nopeammin muistista kuin verkon yli. Jos jostain syystä käyttäjä haluaa varmistaa, että data on ajan tasalla, voi hän tyhjentää välimuistista tiedot ja data silloin haetaan uudestaan APIlta. Lähtevää (POST) dataa ei suoraan tallenneta välimuistiin, sillä API ei välttämättä hyväksy lähetettyä dataa. Portaali päivittää muutokset vain, jos lähetetty data on hyväksytty taustajärjestelmässä.

4.4 Pelimaailman generointi ja renderöinti

Kun portaali on autentikoitunut onnistuneesti ja saanut konfiguraation ladattua, portaali lähettää APIlle ensimmäisen kyselyn, jossa pyydetään galaksijoukkojen dataa. Vastauksena saadaan JSON-taulukko: ”GUID id, int x, int y”, kuten kuvassa 3 näkyy.

Vastauksen klusteri-id:t ovat GUID-arvoja eli Globally unique identifier -arvoja. Ne esitetään 32 merkin heksalukuna esimerkiksi: 6B29FC40-CA47-1067-B31D-00DD010662DA. [33.] Maailman generointi siis tapahtuu klusteri-id:n perusteella, mitä käytetään siemenlukuna klusteri-generaattorille. Klusteri-generaattori palauttaa byte[]-talukon planeettojen indekseistä, ja niiden avulla planeetat saadaan oikeille paikoilleen.

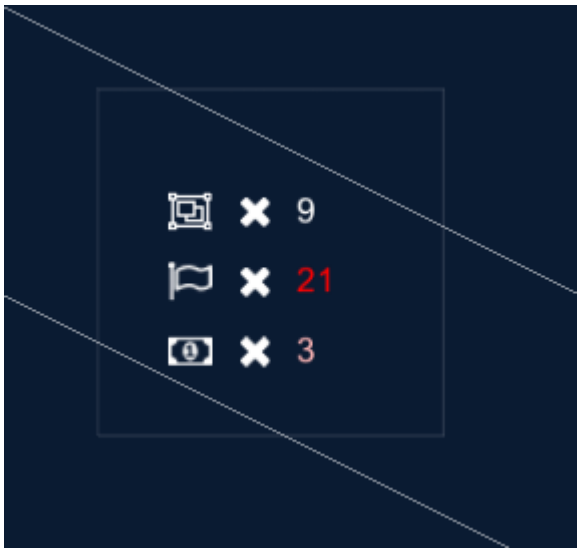
Jokaisen klustereiden id on uniikki, joten maailmasta ei löydy kahta samanlaista klusteria. Pelimaailman proseduaalinen generointi tarkoittaa sitä, ettei pelin asennuspaketissa tarvitse olla koko pelimaailmaa, vaan jokainen pelaaja generoi maailman itse pelin käynnistettäessä. Näin ollen maailmaa on myös helppo muuttaa, kun voidaan vain muuttaa generointialgoritmia tai siemenlukua. Kuvassa 14 pelin konfiguraatiossa nähdään parametreja, joita maailman generaatio käyttää. Siinä voidaan esimerkiksi määritellä, kuinka monta Alliance-planeettaa voi klusteriin maksimissa generoitua.

Portaalin ja pelin maailman esitysmuodot eroavat toisistaan, sillä pelissä pelaaja näkee vain pienen osan maailmasta kerrallaan muutaman planeetan näytöllä. Portaalissa taas on mahdollista zoomata hyvinkin kauas ja nähdä paljon enemmän maailmasta kerralla.

Portaalissa voi siis katsella laajoja alueita, eikä näitä kaikkia alueita voida generoida ja näyttää kerralla. Tähän ongelmaan tehty ratkaisu on jakaa pelimaailman klusterit ryhmittymiin. Aluksi portaaliin tultaessa käyttäjälle näytetään vain klusteriryhmittymiä. Näissä ryhmittymissä planeettoja ei ole vielä generoitu. Klusterit jaetaan ryhmiin niiden x- ja y-koordinaattien perusteella, ottamalla vierekkäin olevat klusterit 3x3-alueelta samaan ryhmään, ja sen jälkeen muodostetaan lisää ryhmiä niin kauan, että kaikki klusterit on saatu johonkin ryhmään. Yhdessä ryhmässä voi olla täten maksimissaan yhdeksän klusteria.

Ryhmissä näytetään kuvan 16 mukaisesti ryhmän klustereiden tietoja. Niillä on tarkoitus saada käyttäjän mielenkiinto herätettyä ja auttaa käyttäjää valitsemaan klusteri, josta hän on kiinnostunut. Esimerkiksi kuvan 16 lipun perässä oleva tieto saadaan seuraavasti: APIlta haetaan samanaikaisesti klustereita ryhmitellessä 50 pelaajien tilit, joilla on eniten ongelmatikettejä. Pelaajien tileiltä palautuu myös tieto, missä klustereissa he ovat, joten ongelmatikettitieto voidaan näyttää klusteriryhmän infossa. Infon tietoja myös värikoodataan liukuvärillä valkoisesta punaiseksi. Mitä punaisempi info on, sitä kriittisempi tilanne on. Väri määräytyy erikseen määritellyn raja-arvon mukaan, mikä helpottaa käyttäjää näkemään nopeasti mahdollisen ongelman vain pelkällä vilauksella.

Näiden tietojen, joita klusteriryhmässä näytetään, pitäisi olla kaikki laskettu valmiiksi taustajärjestelmän puolella. Tällöin voidaan tehdä vain yksinkertaisia kyseleyitä APIlle, kuten pyytää klusterit, joissa on vähiten aktiivisuutta. Näin pystytään katsomaan, onko niissä klustereissa jotain vikaa. Muuten jokainen portaalin käyttäjä joutuisi kyselemään APIlta tuhansien klustereiden pelaajien tietoja ja laskemaan niistä erikseen, mitkä klusterit ovat aktiivisimpia.

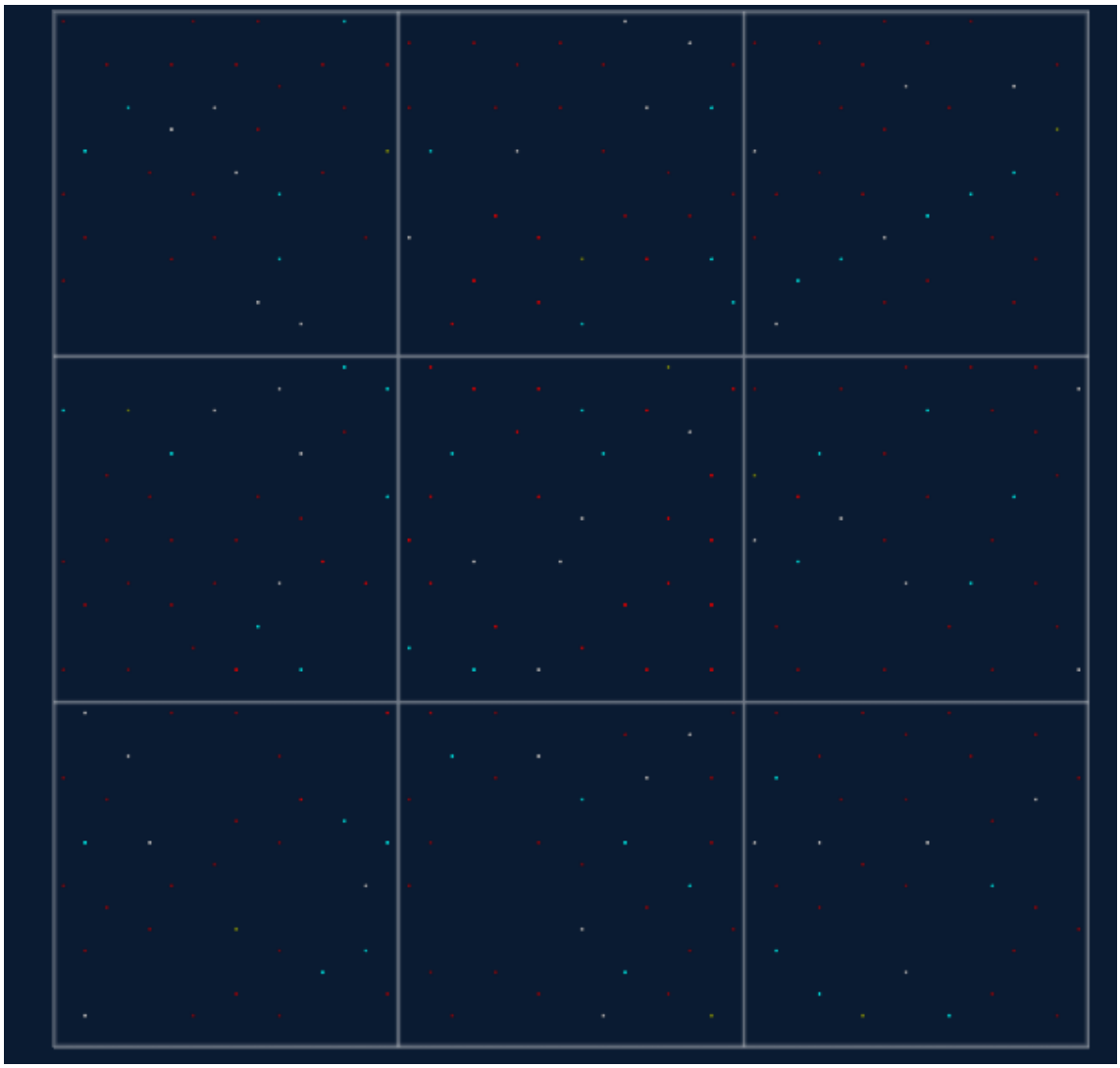


Kuva 16. Klusteriryhmä, jossa klustereiden, ongelmatikettien ja top-100 pelaajien määrä.

Klusteri-näkymä on kaikkein samankaltaisin pelin kanssa. Tässä näkymässä maailma nähdään siinä tilassa, missä se on pelissäkin. Siinä nähdään kaikki yksittäiset planeetat, pelaajat, liittoutumat, tapahtumat ja kaikki muu, mitä pelimaailmassa on.

Klusterit ovat jaettu isompiin 16x16-alueisiin, jotka koostuvat vielä pienemmistä 4x4-alueista. Klusteri koostuu siis 64 * 64 alkion taulukosta. Jokaiselle alkion paikalle voi generoitua planeetta. Kun klusteriryhmään tarkennetaan, generoidaan jokainen 3x3-ryhmittymän klusteri 4.4.1 kappaleen mukaan ja piilotetaan muut klusteriryhmät näkyvistä. Klusteria generoidessa listataan alueet, joihin generoituu iso planeetta, koska nämä alueet voivat muuttua.

Generoinnin jälkeen APIlta käydään kysymässä, mitä listatuista alueista löytyy. Jos alue on tyhjä, silloin ei saada mitään vastaukseksi, mutta jos alueessa on jotain, saadaan vastaukseksi paikkaindeksi 4x4-alueessa, planeetan tyyppi ja kohteen id. Tämän mukaan generoidaan vastauksen palauttaman tyypin planeetat ja linkitetään niihin kohde-Id:t. Lopullinen generaatio näyttää kuvan 17 mukaiselta: siinä on generoituna 9 klusteria ja kaikki isommat planeetat ovat näkyvissä. Ne on värikoodatattu eri värisiksi, jotta ne on nopea ja helppo erottaa toisistaan. Kuvassa 18 on zoomattu kuvan 17 keskimmäisen klusterin vasen yläkulma ja laitettu pelaaja-planeettojen piirtyminen päälle. Kuvassa 19 on sama kohta kuin kuvassa 18, mutta se on taas itse Last Planets-pelistä otettu kuvankaappaus.

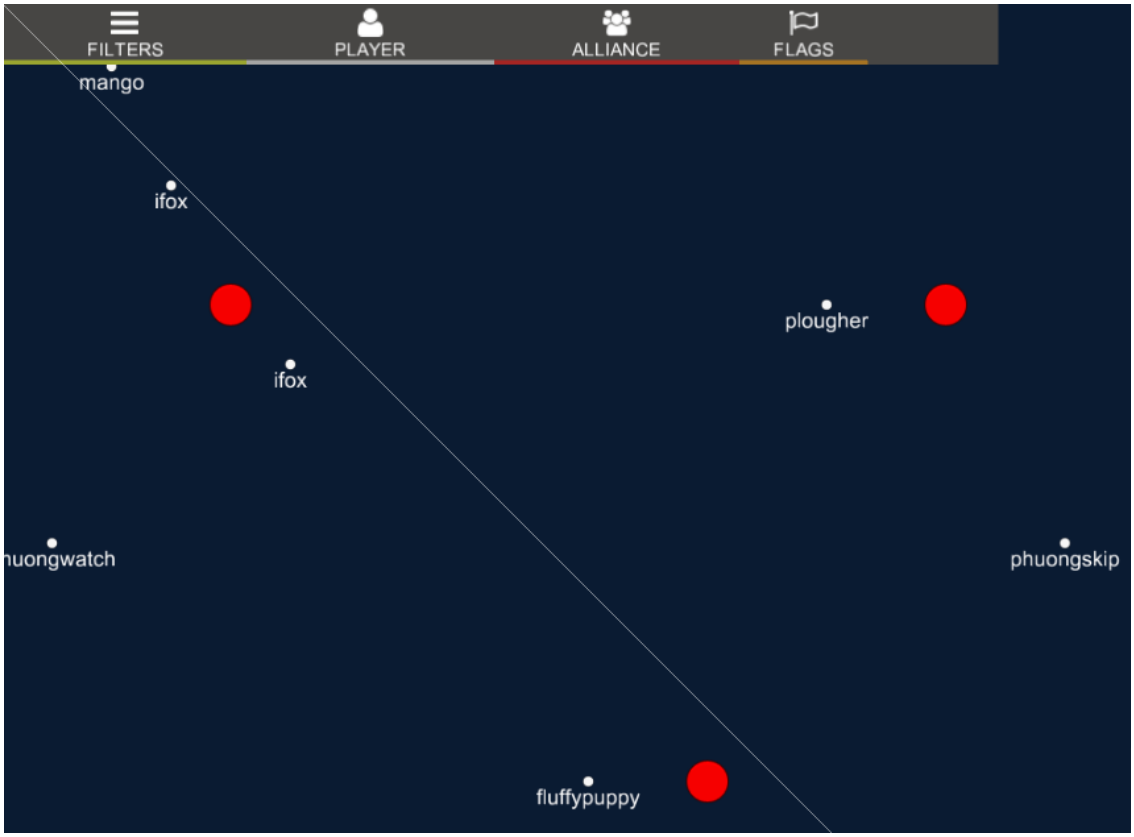


Kuva 17. Yhdeksän klusterin joukko generoituna, pelaajien planeetat on piilotettu.

Jos halutaan tutkia klustereiden pelaajia tarkemmin, pitää portaalin hakea kaikkien loppujen alueiden tiedot APIsta, jolloin saadaan pelaajien id:t linkitettyä pieniin planeettoihin.

Nyt pelaaja- tai mitä tahansa muuta planeettaa klikatessa portaali etsii välimuistista klikatun planeetan kohde-id:llä dataa. Jos dataa ei löydy, tehdään uusi kysely Id:llä planeetan tyypin mukaiseen APIn päätepisteeseen ja tallennetaan välimuistiin APIlta saatu vastaus, jos se on validi. Kun data on saatu joko välimuistista tai hakemalla APIsta, näkymä näyttää tiedot datan pohjalta. Jos pelaajan planeetta painettiin, näytetään pelaajan näkymä. Pelaajan näkymästä voidaan mennä katsomaan pelaajan liitoutumaa liitoutuman Id:llä. Tai voidaan mennä katsomaan pelaajan ongelmatikettejä

niiden Id:llä, jolloin portaali tekee samanlaisen kierroksen kysymällä välimuistista dataa ja niin edelleen.



Kuva 18. Kuvassa on tarkennus kuvan 7 keskimmäisen klusterin vasempaan yläkulmaan. Valkoiset pallot ovat pelaajia ja punaiset liittoutumia.

Kuva 19. Sama kohta klusterista kuin kuvassa 6, mutta otettuna pelistä. Coolkids, Chopshop ja Farmunion ovat liittoutumia ja pienet punaiset pallot bot-planeettoja ja vihreät pelaajia.

4.5 Datan visualisointi

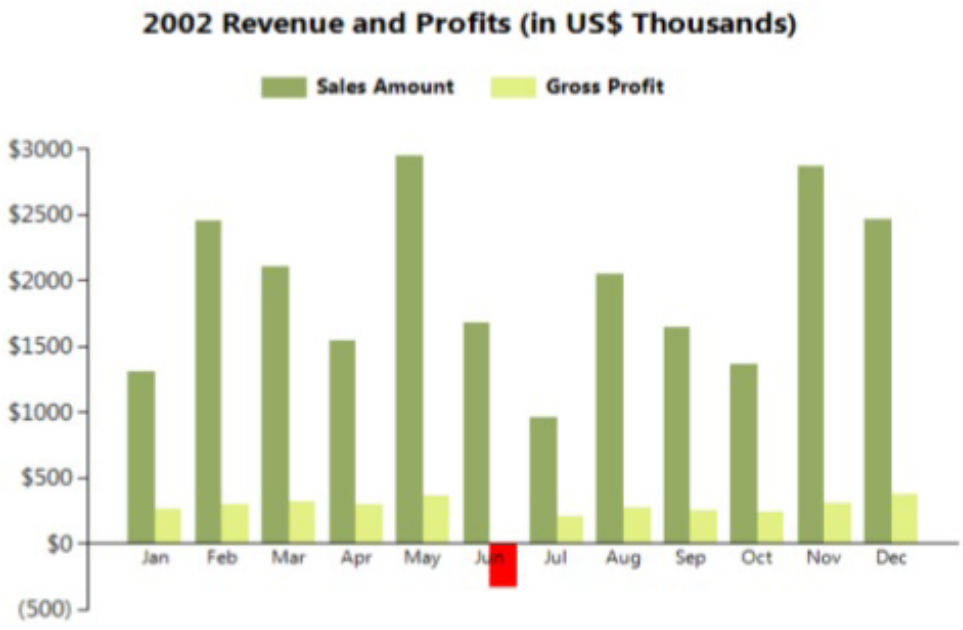
Datan visualisoinnilla portaalissa on tarkoitus auttaa pelin kehittäjiä iteroivassa prosessissa ja päätöksenteossa, kun peliä on tarkoitus päivittää. Peli kerää pelaajilta anonyymisti tietoa Data analytics-palveluun, josta nähdään erilaisia kaavioita sinne kerätyistä tiedoista. Sinne ei kuitenkaan kerätä kaikkea tietoa, mitä tietokannasta löytyy. Portaali on tarkoitettu korkeamman tason datan visualisointiin kuin analytics-palvelu, joten portaalissa pystytään näyttämään enemmän dataa kuin mitä analytics-ohjelmasta on saatavissa ja data voidaan näyttää sellaisena kuin miltä se näyttäisi pelimaailmassa. Dataa voidaan louhia palvelimella tietokannasta, laskea valmiiksi mielenkiintoisia arvoja ja antaa ne APIn kautta portaalin käytettäväksi. Dataa kannattaa louhia, sillä se voi auttaa huijareiden kiinniottamisessa, ekonomian tasapainottamisessa, sillä voidaan laskea tuotantokuluja ja nostaa pelaajien retentiota eli peliin takaisin palaamista. Tästä datasta saadaan varmempaa tietoa kuin

palautekyselyistä, sillä pelaajat eivät aina vastaa todenmukaisesti ja datasta saadaan tietoa joka ikisestä pelaajasta. [34.]

Kuvassa 20 on yksinkertainen esimerkki siitä, miten dataa on helpompi hahmottaa, kun se on ihmiselle sopivassa formaatissa. Tätä myös pyritään soveltamaan portaalissa näytettävissä tiedoissa.

The Value of Data Visualization

Month of Year	Sales Amount	Total Product C...	Gross Profit Ma...	Gross Profit
January	1309863.2511	1046855.0401	0.20079058694...	263008.211
February	2451605.6244	2161789.71439...	0.11821473532...	289815.910000...
March	2099415.6158	1781531.84109...	0.15141536164...	317883.774700...
April	1546592.2292	1250946.0643	0.19115973772...	295646.164900...
May	2942672.90960...	2583467.20809...	0.12206783170...	359205.701500...
June	1678567.4193	2010739.61289...	-0.19789029012...	-332172.193599...
July	962716.741700...	754715.7636	0.21605625942...	208000.978100...
August	2044600.0034	1771778.75389...	0.13343502349...	272821.249500...
September	1639840.109	1393936.67389...	0.14995573882...	245903.43510001
October	1358050.4703	1124337.2647	0.17209463912...	233713.205600...
November	2868129.20330...	2561131.77409...	0.10703751729...	306997.42920002
December	2458472.4342	2085375.78659...	0.15175954076...	373096.647600...



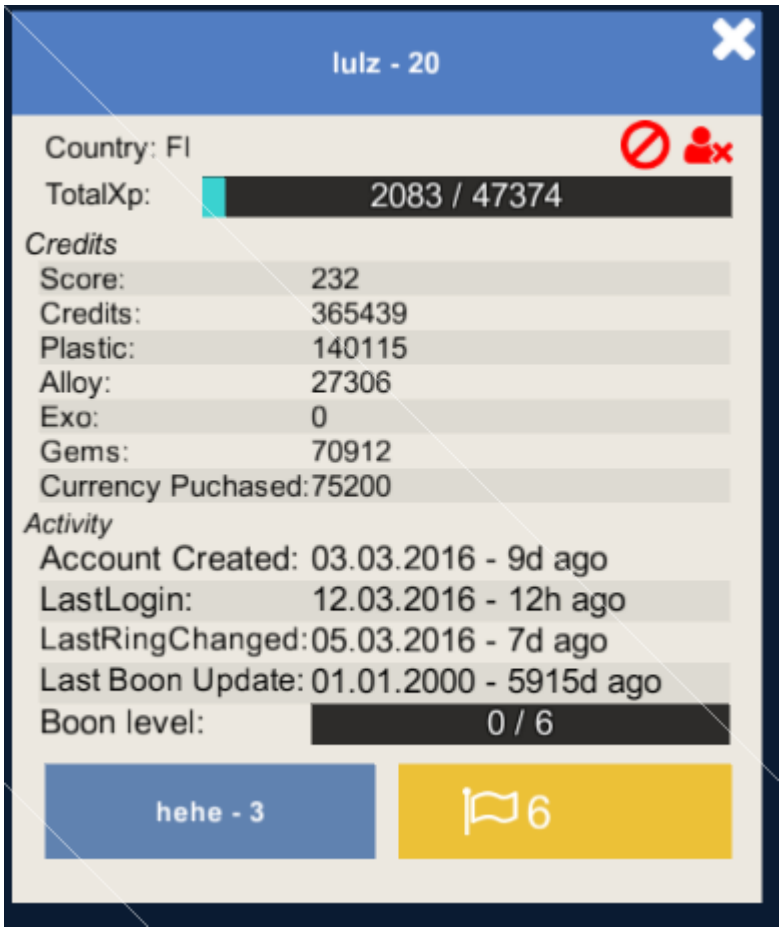
Kuva 20. Data esitettynä eri muodoissa, taulukossa ja pylväsdiagrammeina. [35.]

Erilaisia näkymiä on monia: AccountView, AllianceView, FlagsView ja EventView. Näkymissä näytetään näkymän luokan tiedot. Kaikki näkymät aktivoidaan luokan id:llä tai joukolla id:itä. Id:llä haetaan välimuistista tai APIsta pyydetyn näkymän tiedot. Näkymät lataavat mahdollisimman paljon tiedoista konfiguraatioiden määritelmistä ja luovat valinnat dynaamisesti näkymiin. Silloin näkymiä ei tarvitse erikseen päivittää, riittää vain, että uusin konfiguraatiotiedosto ladataan portaalin käynnistyessä.

Kussakin näkymässä on erilaisia kontrolleja. Esimerkiksi AccountViewissä voidaan antaa tai poistaa pelaajalta porttikielto tai voidaan mennä katsomaan pelaajan liittoumaa tai pelaajan tuottamia ongelmatikettejä.

Muuttujien kohdalla pitää miettiä, miten ne kannattaa näyttää. Ajat muutetaan eurooppalaiseen muotoon dd/mm/yyyy eli päivä, kuukausi ja vuosi. Sen jälkeen aikojen perään lasketaan, kuinka monta päivää kyseisestä ajasta on kulunut. Tämä on helpompi hahmottaa kuin pelkkä päivämäärä. Näkymissä voidaan vielä värikoodata erilaiset arvot helposti, tässä tapauksessa vaikka näyttämällä yli 30 päivää tapahtuneet toiminnot punaisella värillä. Muuttujista, joissa on tiedossa maksimi- ja minimiarvo luodaan palkki. Palkista käyttäjä helposti näkee, kuinka suuri määrä pelaajalla on tiettyä arvoa verrattaessa sen maksimiin.

Kuvassa 21 nähdään esimerkki AccountView-näkymästä. Pelaajan nähdään olevan Lulz ja olevan tasolla 20. Lulz on Suomesta, ja hänelle voidaan antaa porttikielto punaisesta merkistä, jossa on risti henkilön vieressä. Hän on viimeksi kirjautunut sisään 12 tuntia sitten, ja hänellä näyttäisi olevan 6 ongelmatikettiä. Hän kuuluu tason 3 allianssiin jonka nimi on ”hehe”. Näkymästä nähdään kaikki pelaajan tiedot ja voidaan kontrolloida pelaajaa alkeellisesti. Myöhemmin tähän on mahdollista lisätä lisää kontrolleja ja tietoja sitä mukaa kun tarve vaatii.



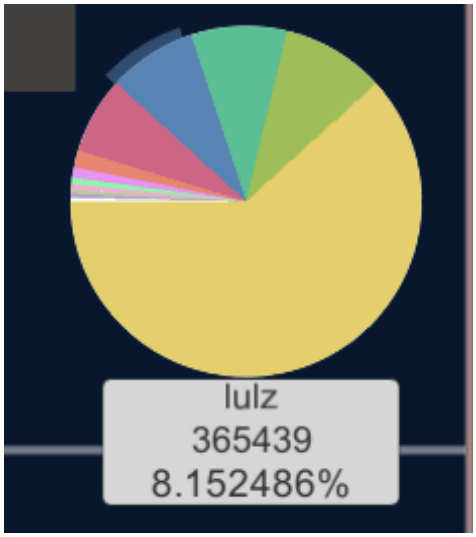
Kuva 21. Pelaajan tiedot -näkymä

Näkymien lisäksi portaalista löytyy ympyrädiagrammi, jolla on helpompi analysoida ja hahmottaa suurempia määriä dataa kuin yksittäisistä näkymistä. Ympyrädiagrammista on helppo erottaa suuruudet osien välillä, joten portaalissa käytetään ympyräkaavioita lämpökartan apuna. Lämpökartasta nähdään, missä päin pelimaailmaa on suurimmat keskittymät jotain tiettyä arvoa, ja ympyrädiagrammi kertoo käyttäjälle, minkä suuruisia nämä arvot ovat kokonaisuudessaan kyseisessä klusteriryhmässä. Kun käyttäjä painaa jotakin palasta ympyrädiagrammista, niin kamera kohdennetaan kartalla arvon omistajaan eli pelaajaan tai allianssiin, ja näin nähdään, mistä kohti tämän arvon omistaja löytyy.

Ympyrädiagrammin arvot saadaan laskettua todella yksinkertaisesti. Lasketaan vain halutun tiedon yhteismäärä klusteriryhmästä ja tämän jälkeen jaetaan yksittäiset arvot tällä yhteismäärällä eli lasketaan prosenttiosuuksia. Aluksi kuitenkin lajitellaan nämä arvot suuruusjärjestykseen ja lasketaan prosenttiosuudet 50 suurimalle arvolle. Loput

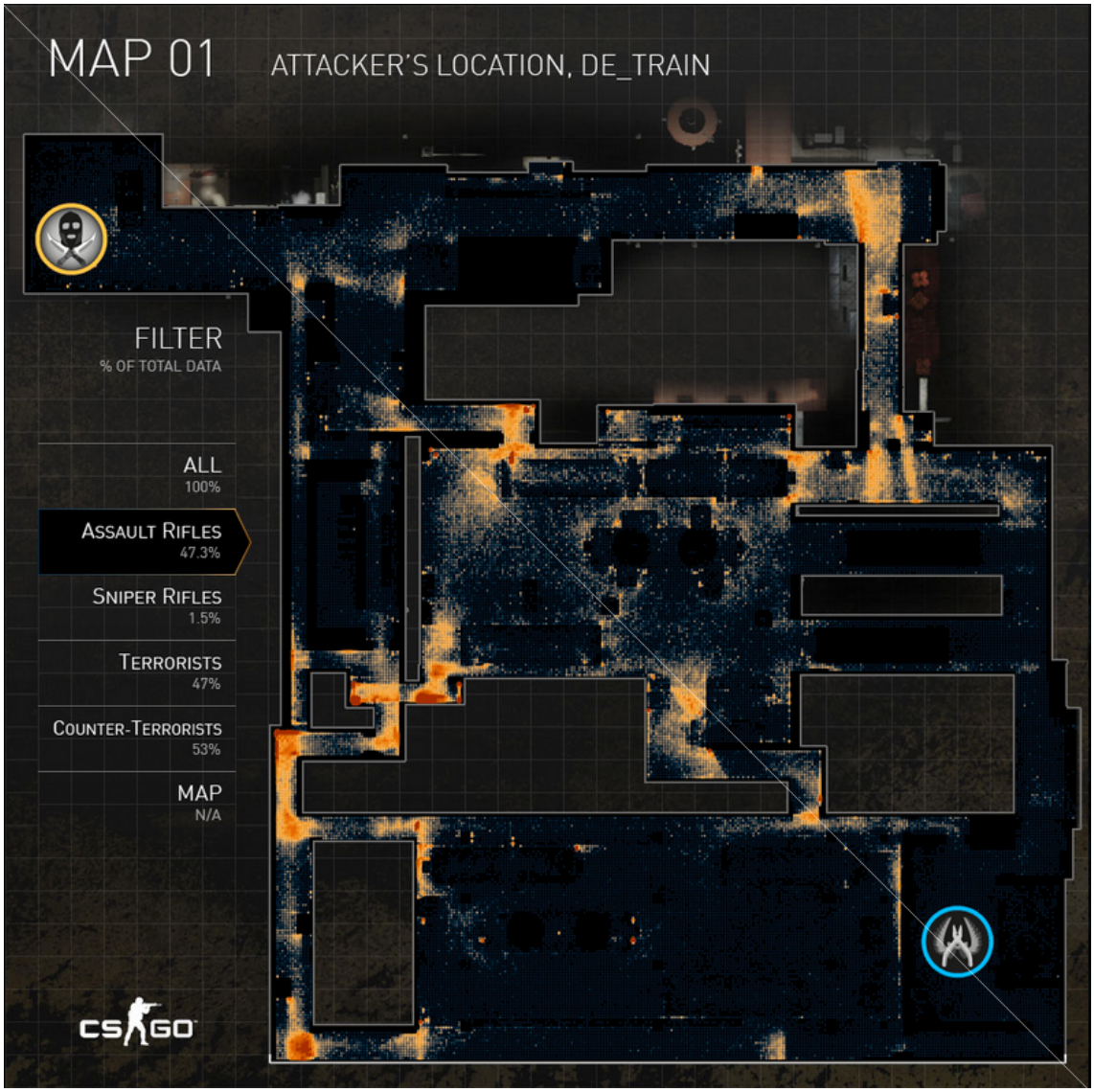
arvot summataan yhteen ja lasketaan niille oma prosenttiosuus, sillä niiden yksittäinen osuus olisi niin pieni, ettei sillä ole käyttäjälle merkitystä, vaan on helpompi nähdä loppujen arvojen yhteenlaskettu osuus.

Ympyrädiagrammin lohkojen ensimmäiset värit on määritelty valmiiksi. Ne ovat miellyttäviä silmälle ja erottuvat hyvin toisistaan. Siitä eteenpäin generoidaan värejä niin monta kuin on tarvetta. Värit generoidaan ennalta määritellyn siemenluvun pohjalta, jolloin alkion x väri pysyy aina samana. Tällä värigeneraatiolla saadaan toisistaan hyvin eroavia värejä. Lopullinen ympyrädiagrammi voidaan nähdä kuvassa 22.



Kuva 22. Ympyrädiagrammi, jossa valittuna on sininen osa. Laatikossa on lueteltuna pelaaja jolle tämä osa kuuluu, osan arvo ja prosenttimäärä kokonaisuudesta.

Datan visualisointiin on erilaisia toteutuksia ympyrädiagrammin lisäksi. Näistä yksi on lämpökartta. Lämpökartta on kartta, joka koostuu erivärisistä pisteistä koordinaatistossa. Lämmin väri kuvaa suurta tiheyttä, ja kylmä väri taas pienempää tiheyttä. Lämpökartalla on hyvä visualisoida asioiden esiintymistiheyksiä jollakin alueella. Kuvassa 23 on hyvä esimerkki lämpökartasta. Lämpökartta on Counter Strike Global Offensive -pelistä, ja siitä nähdään, missä kohdin karttaa pelaajat ampuvat eniten laukauksia. Tällaisten visualisaatioiden kautta on kenttäsuunnittelijoiden paljon helpompi tehdä kartasta miellyttävämpi kokemus pelaajille.



Kuva 23. Lämpokartta Counter Strike Global Offensive -pelistä. Punaiset pisteet näyttävät, missä kohdin karttaa on ammuttu eniten. Tämän perusteella karttoja on helpompi tasapainottaa. [36.]

Lämpökartat usein koostuvat liukuväristä: sinisestä, vihreästä ja punaisesta. Se toimii hyvin tapauksissa, joissa koordinaatit voivat olla missä vain. Tässä tapauksessa koordinaatit voivat olla vain planeettojen paikkakoordinaateissa eli 64x64-taulukossa. Ne eivät voi siis olla niin lähekkäin toisiaan, että piirretyt pisteet olisivat limittäin. Tämän takia tultiin siihen tulokseen, että silloin parhaiten toimii liukuväri sininen – punainen.

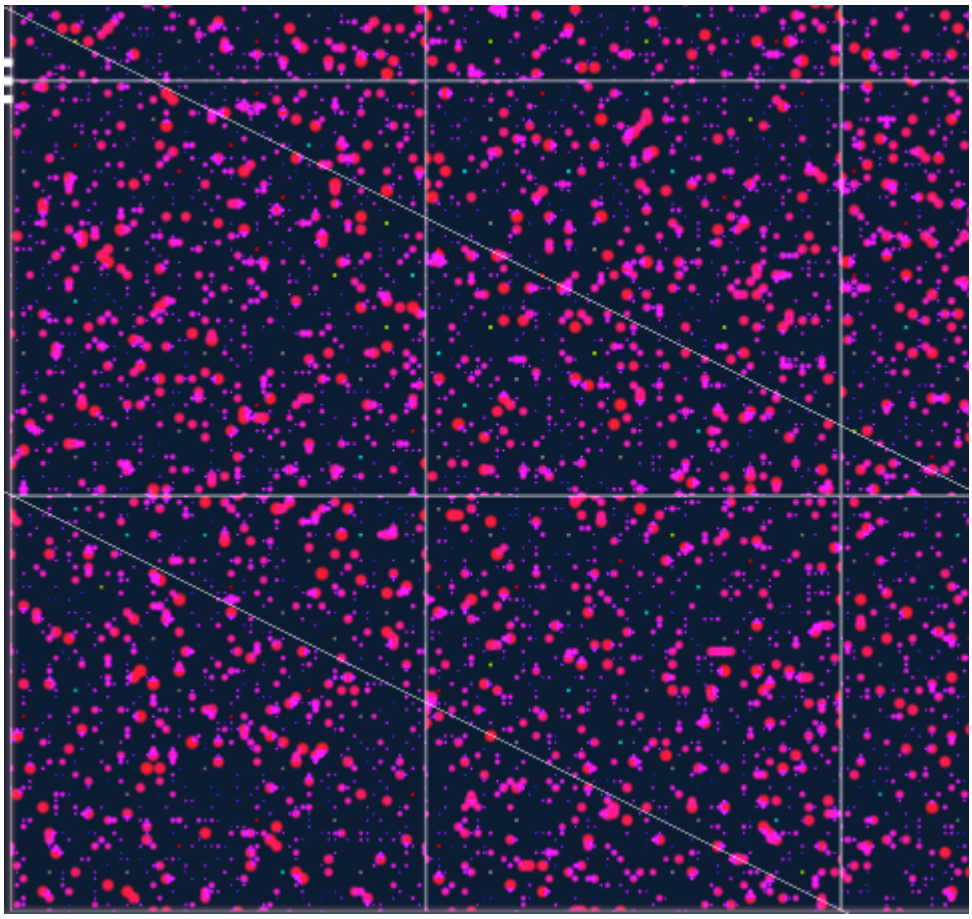
Lämpökartan data saadaan hakemalla välimuistista esimerkiksi klusterin kaikki pelaajat listaan List<Player>. Oletetaan että halutaan kuvata pelaajia, joilla on maailmassa

eniten metallia pankissa. Etsitään listasta arvo alloy.Max(). Nyt portaalissa on lämpökartalle rajattu arvot, minimi on 0 ja maksimi on suurin löydetty alloy-arvo. Tämän jälkeen normalisoidaan arvot välille 0 – 1.

$$value = (value - min) / (max - min)$$

Tämän jälkeen ollaan valmiita piirtämään lämpökartta. Käydään kaikki pelaajat läpi ja piirretään pelaajan koordinaattien kohdalle valkoinen piste, jonka alphaväri on pelaajan alloy normalisoituna minimi- ja maksimiarvon välille. Pisteen säde määräytyy myös normalisoidun arvon mukaan, jolloin pieni arvo jää pieneksi ja siniseksi, kun taas arvo joka on lähellä suurinta, tulee isoksi ja punaiseksi.

Kun kaikki valkoiset pisteet on piirretty, käydään tekstuuri läpi ja muutetaan värit lämpökartalle valitun liukuvärin mukaan. Väri määräytyy liukuväristä alpha-arvon mukaan. Tämän jälkeen meillä on tekstuuri lämpökarttaan. Liitetään tekstuuri varjostimeen ja laitetaan se 3D-maailmassa olevaan tasoon, joka on samassa kohdassa klusterin kanssa. Lopputulos on kuvan 24 näköinen.



Kuva 24. Lämpökartta neljästä klusterista. Suuret punaiset pisteet ovat pelaajia, joilla on eniten metallia, sinisillä taas vähiten. Klusterit on täytetty testidatalla.

5 Tulokset ja jatkokehitys

5.1 Tulokset

Portaali on siinä vaiheessa, että se on juuri siirtymässä tuotantopalvelimelle käyttöön. Alustava vastaanotto on ollut Vulpine Gamesilta hyvinkin positiivista. Portaali toimii ja siitä pystyy hallinnoimaan pelimaailmaa osittain. Portaalista voi tällä hetkellä antaa ja poistaa pelaajilta porttikieltoja, luoda uusia tapahtumia manuaalisesti ja ratkaista ongelmatikettejä. Koko pelimaailmaa pystyy tarkkailemaan, ja käyttäjä pystyy seuraamaan klustereiden pelaajia sekä liittoutumien kehitystä. Tietojen seuraamiseen portaalista löytyy erilaisia näkymiä, jotka näyttävät tiedot helposti havaittavassa muodossa, ja lisäksi löytyy esimerkiksi ympyrädiagrammi, jolla voidaan analysoida arvojen osuuksia toistaan klusterissa. Käyttäjä voi myös luoda lämpökarttoja klusteriryhmissä, havainnollistamaan, keitä pelaajia vastaan on hyökätty lähiaikoina tai ketkä ovat kirjautuneet viimeksi peliin ja näin ollen käyttäjä pääsee tarkkailemaan, onko klusterin sijainnilla tai hyökkäysten määrällä väliä esimerkiksi pelaajien aktiivisuuteen. Tämän pitäisi tuoda paljon lisäarvoa pelin kehitykseen ja päätösten tekemiseen. Portaalia ei kuitenkaan ole käytetty vielä tuotantodatalla, ja voi olla, että portaalista saattaa puuttua joitakin hyvinkin tärkeitä työkaluja, joten portaalin kehittämiseen kannattaa panostaa jos peli lähtee hyvin käyntiin esijulkaisussa.

5.2 Jatkokehitys

Tällä hetkellä portaali on vain WebGL-sovellus, joka tarkoittaa sitä, että samalle nettisivulle, jonne portaali on upotettu, on mahdollista lisätä muita hallintatyökaluja. Näiden työkalujen ei tarvitse kuulua WebGL-sovellukseen, vaan ne voivat olla omia pieniä sovelluksiaan portaalin rinnalla.

Kun peliin tulee uusia suuria ominaisuuksia, esimerkiksi uusia planeettoja tai jos pelaajat saavat uusia toimintoja, joilla voivat vaikuttaa enemmän pelimaailmaan, niiden lisäämisestä ja päivittämisestä portaaliin täytyy jatkossa huolehtia.

Tuotantodatalla ei ole vielä pystytty tekemään pidempiaikaista testaamista, joten portaalista varmasti löytyy jonkin verran bugeja. Myöskään suurella määrällä tuotantodataa ei ole päästy testaamaan, joten saa nähdä, tarvitaanko vielä suurempaa yleiskatsaustasoa klusteriryhmistä. Klusteriryhmä-näkymiin pitää jatkossa laskea lisää joitain merkittäviä arvoja käyttäjille, jotta käyttäjä osaa jo tässä vaiheessa päättää mitä klusteriryhmää hän haluaa tutkia tarkemmin. Myöskin tähän näkymään olisi hyvä saada erinäisiä datan visualisointi mahdollisuuksia. Olisi hyvä päästä näkemään, missä klusteriryhmässä on eniten aktiivisuutta ja missä taas vähiten.

Koontiversion kääntäminen automaattisesti uuden kommitin jälkeen ja testien automatisointi olisi myös hyvä tehdä, se vähentäisi manuaalista työtä. WebGL:n koonti prosessi on todella hidas sen monimutkaisuuden ja monen eri prosessin takia, joten olisi todella kätevää, että esimerkiksi Jenkins kävisi git:stä hakemassa master-haaran uusimman kommitin, ajaisi testit ja niiden onnistuessa kokoaisi ja asentaisi tuotantoon uusimman version portaalista, jolloin portaalin testaaminen nopeutuisi.

Kuten aiemmin mainittiin, demojen katsominen olisi hyvä integroida portaaliin, jolloin ongelmatikettien ratkaiseminen helpottuisi. Etenkin jos tiketti on tullut ongelmasta taistelussa, voitaisiin nähdä, mitä kummallista on tapahtunut. Lisää ominaisuuksia, joita voitaisiin portaaliin kenties liittää, olisivat palvelimien error-viestien katselmointi portaalista, jolloin portaaliin olisi keskitettynä enemmän asioita.

Portaalille tarjottu API ei tällä hetkellä tarjoa lähes yhtään valmiiksi louhittua dataa tietokannasta. Big Datan kanssa on todella paljon mahdollisuuksia saada portaalin vielä enemmän merkittäviä analysointimahdollisuuksia, kuten kappaleessa 4.6 selitettiin. Tämän datan hyödyntäminen on todella tärkeää pelin menestyksen kannalta. Pelin käyttämään game analytics -sovellukseen on tallennettu jotain tietoa, jota ei pelin tietokannasta löydy ja toisinpäin. Analytics-palvelusta nähdään erilaisia graafeja sinne tallennetusta datasta, mutta sieltä ei löydy kaikkea dataa, eikä sitä saa visualisoitua samaan tapaan kuin portaalissa. Analytics-palvelusta voisi saada tietoa pelaajien iästä, sukupuolesta, laitteesta, jota he käyttävät ja niin edelleen. Sieltä olisi siis mahdollista saada paljon merkittävää dataa, yhdistää ne tietokannan kanssa ja näyttää ne portaalissa yhdessä. Kaiken tämän lisäksi portaalissa on vain juuri tämän hetken tilanne. Olisi mielenkiintoista saada portaalin myös vanhempaa dataa ja näin ollen

verrattua niitä. Tämä olisi aika tärkeää suunnittelun kannalta, nähtäisiin ollaanko menossa oikeaan suuntaan.

Myöskin Unityn uusia päivityksiä stabiilihaarassa pitää seurata, sillä portaalin käyttöliittymässä on tällä hetkellä pari ohjelmointivirhettä, joita ei ole vielä saatu korjattua Unityn puolelta. WebGL:ään on tulossa myös parannuksia, esimerkiksi juuri WebGL:n käännösajan pitäisi nopeutua huimasti Unityn seuraavassa versiossa. [37.] Portaaliin myös jää keskeneräiseksi pari kohtaa, koska taustajärjestelmän API on vielä hieman keskeneräinen.

6 Yhteenveto

Opinnäytetyön tavoitteena oli luoda hallintaportaali, jolla voidaan hallinnoida ja seurata Last Planets-nimisen mobiilipelin pelimaailmaa. Työ aloitettiin suunnittelulla ja portaalin haluttujen ominaisuuksien määrittelyllä. Toteutus eteni suoraviivaisesti, ja ensimmäiset ongelmat ilmenivät, kun Unitystä ruvettiin kääntämään projektia WebGL-versioksi. Projektin sisäiseen koodipohjaan piti tehdä muutoksia, jotta projekti ensinnäkin suostui kääntymään. Myös testausvaiheessa löytyi pari ongelmaa WebGL-versiosta, jotka piti ratkaista. Nämä johtuivat ominaisuuksista, jotka toimivat C#:lla, mutta eivät Javascriptissä.

API, jota portaali käyttää, on todella hyvä, sillä vaikka taustajärjestelmän puolella tehtiin muutoksia, tarjosi määritelty API silti samanlaiset tulokset, jolloin portaalia ei tarvinnut muokata. Pieniä muutoksia portaaliin ominaisuuksiin on kuitenkin tehty kehitystyön aikana, koska peli on myös kehitystyön alla, jolloin jotkin ominaisuudet muuttuivat niin, että portaalin piti myös mukautua.

Työssä saavutettiin kaikki asetetut tavoitteet ja osittain enemmänkin toiminnallisuuksia kuin alkuperäisessä määrittelydokumentissa oli listattuna. Alkuperäisessä määrittelydokumentissa oli määriteltynä pelimaailman seuraaminen eli mihin planeetat generoituvat ja miltä maailma näyttää. Sen lisäksi pelaajien, liittoutumien ja ongelmatikettien seuraaminen sekä niiden yksinkertainen kontrollointi. Esimerkiksi ympyrädiagrammi toteutettiin auttamaan datanvisualisoinnissa, koska sen

toteuttamisen huomattiin olevan helppoa ja tuovan paljon lisäarvoa portaaliin, mutta sitä ei ollut alkuperäisessä spesifikaatiossa.

Nähtäväksi jää, miten portaali toimii tuotantodatalla, kun peli esijulkaistaan, ja mitkä tulevat olemaan tärkeitä ominaisuuksia. Nähtäväksi jää myös, tarvitseeko jotain ominaisuuksia kenties parantaa tai puuttuuko portaalista jotain ominaisuuksia, jotka olisivat avuliaita maailman moderoinnissa.

Unityn WebGL-tuki on vielä suhteellisen nuori ja sen kanssa on omat haasteensa. Javascriptiin ollaan lisäämässä uusia ominaisuuksia kuten threading, joten tulevaisuudessa WebGL tulee olemaan vielä parempi. Hyvä puoli on, ettei WebGL tarvitse mitään lisäosia selaimissa ja se voi tulla toimimaan hyvin puhelimissakin tulevaisuudessa. Myöskin asm.js Javascriptin suorituskyky tulee vielä tulevaisuudessa nousemaan, jolloin WebGL-sovelluksista saadaan suorituskykyisempiä. [38.]

Lähteet

1 Mark Robinson 2013 Game development conference: Why players are leaving your game. Verkkodokumentti. <http://www.gdcvault.com/play/1020101/Why-Players-are-Leaving-Your>. Luettu 22.4.2015.

2 Eve online fanisivu. Verkkodokumentti. <http://evemaps.dotlan.net/universe>. Luettu 23.4.2015.

3 AWS Invent 2015 konferenssin Netflix insinöörin puhe. Video. <https://www.youtube.com/watch?v=-mL3zT1iIKw>. Luettu 23.4.2015.

4 Hypertext Transfer Protocol – HTTP/1.1 spesifikaatio. Verkkodokumentti. <https://tools.ietf.org/html/rfc2616#section-4>. Luettu 22.4.2015.

5 JSON kotisivu. Verkkodokumentti. <http://www.json.org/>. Luettu 22.4.2015.

6 JSON standardi. Verkkodokumentti <http://www.ecma-international.org/publications/files/ECMA-ST/ECMA-404.pdf>. Luettu 22.4.2015.

7 JSON serialization / deserilization kirjasto C# kielelle. Verkkodokumentti. <https://github.com/jacobdufault/fullserializer>. Luettu 22.4.2015.

8 OAuth standardin kotisivu Verkkodokumentti <http://oauth.net/2/>. Luettu 22.4.2015.

9 OAuth 2.0 tutoriaali Verkkodokumentti <http://docs.apigee.com/api-services/content/oauth-20-client-credentials-grant-type>. Luettu 22.4.2015.

10 Microsoftin Azure palvelun autentikaation dokumentointi. Verkkodokumentti <https://msdn.microsoft.com/en-us/library/azure/dn645543.aspx>. Luettu 22.4.2015.

11 Microsoftin verkko palveluiden dokumentaatio. Verkkodokumentti <https://msdn.microsoft.com/en-us/library/ff512387.aspx>. Luettu 22.4.2015.

12 Git dokumentaatio submoduuleista. Verkkodokumentti <https://git-scm.com/book/en/v2/Git-Tools-Submodules>. Luettu 22.4.2015.

13 Unity3D suhdetoiminta kotisivu. Verkkodokumentti <https://unity3d.com/public-relations>. Luettu 22.4.2015.

14 C#-ohjelmointikielen verkkokurssi. Verkkodokumentti <http://www.cprogramming.com/tutorial/csharp.html>. Luettu 22.4.2015.

15 Khronos yhdistyksen kotisivu. Verkkodokumentti <https://www.khronos.org/webgl/>. Luettu 22.4.2015.

16 Diaesitys Unitystä ja WebGLstä. Verkkodokumentti. <http://www.slideshare.net/LiorTal1/lessons-learned-with-unity-and-webgl>. Luettu 23.4.2015.

17 Unity3D blogikirjoitus webgllstä. Verkkodokumentti <http://blogs.unity3d.com/2014/04/29/on-the-future-of-web-publishing-in-unity/>. Luettu 23.4.2015.

18 Monoprojektin kotisivu. Verkkodokumentti <http://www.mono-project.com/>. Luettu 23.4.2015.

19 Unity3D blogikirjoitus tulevaisuuden skriptauksesta. Verkkodokumentti. <http://blogs.unity3d.com/2014/05/20/the-future-of-scripting-in-unity/> . Luettu 23.4.2015.

20 Diaesitys Emscripten-kääntäjästä. Verkkodokumentti. <http://kripken.github.io/mloc_emscripten_talk/#/11>. Luettu 23.4.2015.

21 Emscriptenin kotisivu. Verkkodokumentti. <http://kripken.github.io/emscripten-site/>. Luettu 23.4.2015.

22 Asm.js usein kysytyt kysymykset. Verkkodokumentti. <http://asmjs.org/faq.html>. Luettu 23.4.2015.

23 Mozillan blogikirjoitus asm.js kielestä ja sen suorituskyvystä. Verkkodokumentti. <https://blog.mozilla.org/luke/2014/01/14/asm-js-aot-compilation-and-startup-performance/>. Luettu 23.4.2015.

24 Unity3D WebGL-dokumentaatio suorituskyvystä. Verkkodokumentti. <http://docs.unity3d.com/Manual/webgl-performance.html>. Luettu 23.4.2015.

25 Unity3D WebGL etenemissuunnitelma. Verkkodokumentti. <http://forum.unity3d.com/threads/webgl-roadmap.334408/>. Luettu 23.4.2015.

26 NUnit C#-testauskehikon kotisivu ja dokumentaatio. Verkkodokumentti. <http://www.nunit.org/>. Luettu 23.4.2015.

27 Unity3Dn dokumentaatio WebGLn kääntämisprosessista. Verkkodokumentti. <http://docs.unity3d.com/Manual/webgl-building.html>. Luettu 23.4.2015.

28 Apachen kotisivu. Verkkodokumentti. <http://www.apache.org/>. Luettu 23.4.2015.

29 Stackshare.ion 2015 vuoden lista suosituimmista sovelluskehitys-työkaluista. Verkkodokumentti. <http://stackshare.io/posts/top-50-developer-tools-and-services-of-2015#top-50>. Luettu 23.4.2015.

30 Rediksen kotisivu. Verkkodokumentti. <http://redis.io/topics/introduction>. Luettu 23.4.2015.

31 Soap- ja Rest-protokollien vertailu. Verkkodokumentti. <http://spf13.com/post/soap-vs-rest>. Luettu 23.4.2015.

32 Font Awesomen kotisivu. Verkkodokumentti. <https://github.com/FortAwesome/Font-Awesome>. Luettu 23.4.2015.

33 Microsoftin .NET-frameworkin dokumentaatio. Verkkodokumentti. <https://msdn.microsoft.com/en-us/library/aa373931%28VS.85%29.aspx>. Luettu 23.4.2015.

34 Blogikirjoitus miten data auttaa pelin suunnittelussa. Verkkodokumentti. <http://www.gamasutra.com/view/feature/131225/better_game_design_through_data_.php?page=1>. Luettu 23.4.2015.

35 Data visualisaation parhaat käytännöt. Verkkodokumentti. <http://www.slideshare.net/idigdata/data-visualization-best-practices-2013/7-Gestalt_Principles_of_Perception_Objects>. Luettu 23.4.2015.

36 CSGO:n lämpökartta. Verkkodokumentti. <http://blog.counter-strike.net/science/maps.html>. Luettu 23.4.2015.

37 Unity3D blogikirjoitus 5.4 version parannuksista. Verkkodokumentti. <http://blogs.unity3d.com/2016/03/15/enhanced-visuals-better-performance-and-more-the-unity-5-4-public-beta-is-ready/>. Luettu 23.4.2015.

38 Mozillan blogikirjoitus teknologian etenemissuunnitelmasta. Verkkodokumentti. <https://blog.mozilla.org/futurereleases/2015/07/02/mozilla-games-technology-roadmap/>. Luettu 23.4.2015.